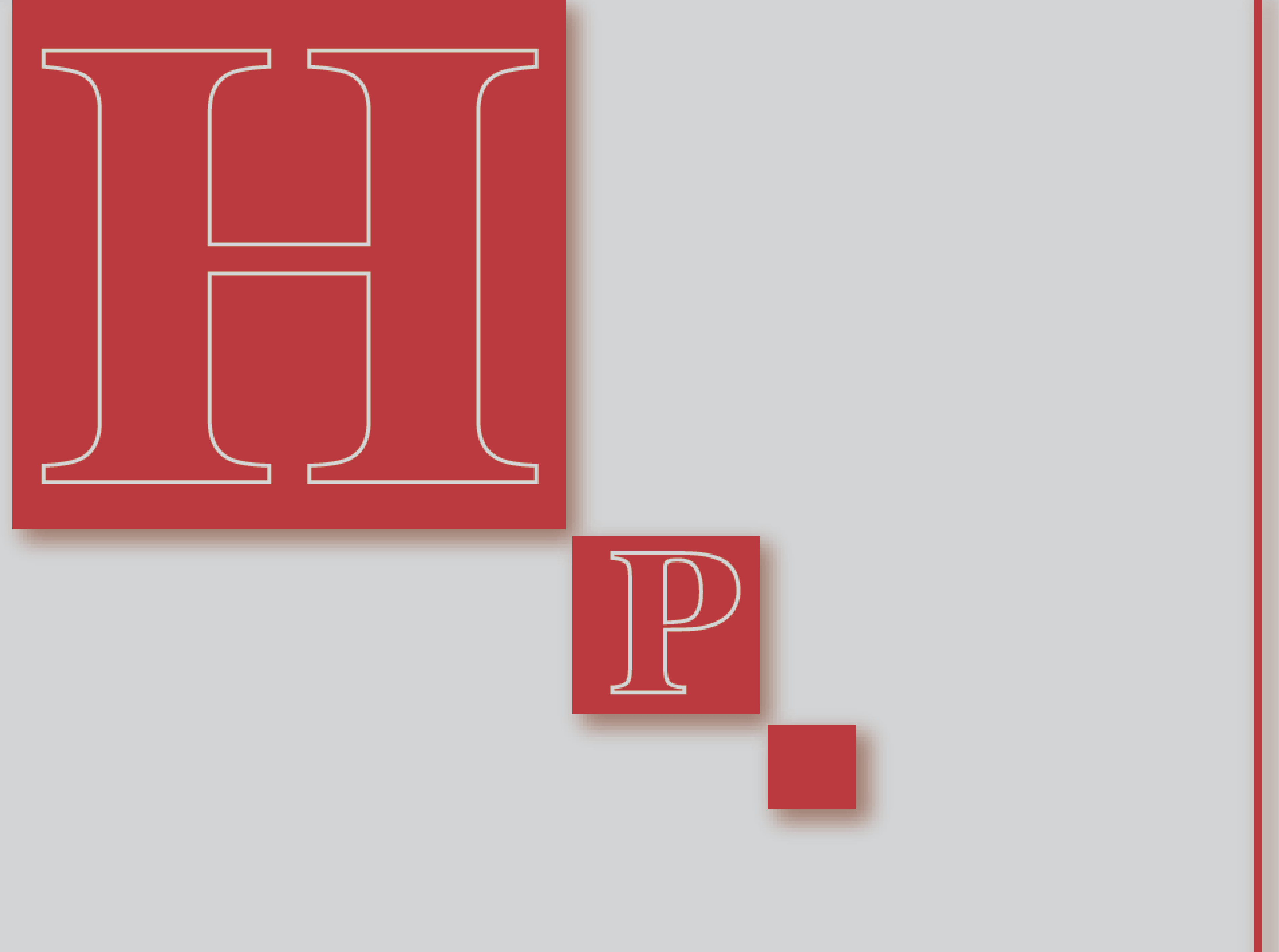
21世纪高等学校计算机基础实用规划教材

PHP Web 程序设计教程与实验



徐　辉　主编
卢守东　蒋曹清　副主编



清华大学出版社

21世纪高等学校计算机教育实用规划教材

PHP Web 程序设计教程与实验

徐　辉　主编
卢守东　蒋曹清　副主编

清华大学出版社
北　京

内 容 简 介

本书以 PHP 5 为主要编程工具，以 MySQL 4.1 为 Web 数据库，结合多年的 Web 网站开发的实际经验和教学，全面地介绍开发 Web 动态网页的程序设计技术，内容包括 Web 基础、HTML 基础、JavaScript 客户端脚本语言、动态 Web 网站环境的构建、PHP 语法基础、PHP 面向对象编程、MySQL 数据库操作、PHP 访问 MySQL 数据库、会话和用户认证、XML 语法、PHP 5 的 XML 文档解析、基于 PHP 5 的 Web 服务、XML 与数据库之间数据交换、网络考试应用系统实例等内容。

本书内容丰富，结构合理，由浅入深，例题丰富，实验内容充实，实用性强，每章后面安排有习题和实验题。

本书可作为高等院校本科或专科计算机、电子商务、信息管理与信息系统以及相关专业的 Web 开发课程的教材或教学参考书，也可供 PHP 编程人员参考。

图书在版编目（CIP）数据

PHP Web 程序设计教程与实验/徐辉主编. —北京：清华大学出版社，2008.1
（21 世纪高等学校计算机教育实用规划教材）
ISBN 978-7-302-15550-8

Ⅰ. P…　Ⅱ. 徐…　Ⅲ. PHP 语言－程序设计－高等学校－教材　Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2007）第 096728 号

责任编辑：魏江江　徐跃进
责任校对：李建庄
责任印制：

出版发行：清华大学出版社　　**地　　址：**北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编：**100084
c-service@tup.tsinghua.edu.cn
社 总 机：010-62770175　　**邮购热线：**010-62786544
投稿咨询：010-62772015　　**客户服务：**010-62776969
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：185×260　**印　张：**28.25　　**字　数：**681 千字
版　　次：2008 年 1 月第 1 版　　**印　次：**2008 年 1 月第 1 次印刷
印　　数：1~ 000
定　　价：　　元

本书如存在文字不清、漏印、缺页、倒页、脱页等印装质量问题，请与清华大学出版社出版部联系调换。联系电话：010-62770177 转 3103　产品编号：

出版说明

随着我国高等教育规模的扩大以及产业结构调整的进一步完善，社会对高层次应用型人才的需求将更加迫切。各地高校紧密结合地方经济建设发展需要，科学运用市场调节机制，合理调整和配置教育资源，在改革和改造传统学科专业的基础上，加强工程型和应用型学科专业建设，积极设置主要面向地方支柱产业、高新技术产业、服务业的工程型和应用型学科专业，积极为地方经济建设输送各类应用型人才。各高校加大了使用信息科学等现代科学技术提升、改造传统学科专业的力度，从而实现传统学科专业向工程型和应用型学科专业的发展与转变。在发挥传统学科专业师资力量强、办学经验丰富、教学资源充裕等优势的同时，不断更新教学内容、改革课程体系，使工程型和应用型学科专业教育与经济建设相适应。计算机课程教学在从传统学科向工程型和应用型学科转变中起着至关重要的作用，工程型和应用型学科专业中的计算机课程设置、内容体系和教学手段及方法等也具有不同于传统学科的鲜明特点。

为了配合高校工程型和应用型学科专业的建设和发展，急需出版一批内容新、体系新、方法新、手段新的高水平计算机课程教材。目前，工程型和应用型学科专业计算机课程教材的建设工作仍滞后于教学改革的实践，如现有的计算机教材中有不少内容陈旧（依然用传统专业计算机教材代替工程型和应用型学科专业教材），重理论、轻实践，不能满足新的教学计划、课程设置的需要；一些课程的教材可供选择的品种太少；一些基础课的教材虽然品种较多，但低水平重复严重；有些教材内容庞杂，书越编越厚；专业课教材、教学辅助教材及教学参考书短缺，等等，都不利于学生能力的提高和素质的培养。为此，在教育部相关教学指导委员会专家的指导和建议下，清华大学出版社组织出版本系列教材，以满足工程型和应用型学科专业计算机课程教学的需要。本系列教材在规划过程中体现了如下一些基本原则和特点。

（1）面向工程型与应用型学科专业，强调计算机在各专业中的应用。教材内容坚持基本理论适度，反映基本理论和原理的综合应用，强调实践和应用环节。

（2）反映教学需要，促进教学发展。教材规划以新的工程型和应用型专业目录为依据。教材要适应多样化的教学需要，正确把握教学内容和课程体系的改革方向，在选择教材内容和编写体系时注意体现素质教育、创新能力与实践能力的培养，为学生知识、能力、素质协调发展创造条件。

（3）实施精品战略，突出重点，保证质量。规划教材建设仍然把重点放在公共基础课和专业基础课的教材建设上；特别注意选择并安排一部分原来基础比较好的优秀教材或讲义修订再版，逐步形成精品教材；提倡并鼓励编写体现工程型和应用型专业教学内容和课程体系改革成果的教材。

（4）主张一纲多本，合理配套。基础课和专业基础课教材要配套，同一门课程可以有

多本具有不同内容特点的教材。处理好教材统一性与多样化，基本教材与辅助教材、教学参考书，文字教材与软件教材的关系，实现教材系列资源配套。

（5）依靠专家，择优选用。在制订教材规划时要依靠各课程专家在调查研究本课程教材建设现状的基础上提出规划选题。在落实主编人选时，要引入竞争机制，通过申报、评审确定主编。书稿完成后要认真实行审稿程序，确保出书质量。

繁荣教材出版事业，提高教材质量的关键是教师。建立一支高水平的以老带新的教材编写队伍才能保证教材的编写质量和建设力度，希望有志于教材建设的教师能够加入到我们的编写队伍中来。

21 世纪高等学校计算机教育实用规划教材编委会

联系人：丁岭 dingl@tup.tsinghua.edu.cn

前 言

PHP 是目前最流行的 Web 服务器端编程语言之一，能够根据用户请求或者服务器端的数据生成动态网页。PHP 是一种跨平台的 HTML 内嵌式语言，可以在 Windows、UNIX 等不同平台上运行，程序移植方便。

MySQL 是一个多用户、多线程的数据库服务器，支持 SQL 标准化语言。PHP 和 MySQL 都是简单易学、运行速度快且功能强大的免费软件包，两者成为构建动态 Web 网站的强有力组合，近年来被越来越多地应用于 Web 网站的建设中。据 Netcraft 公司的不完全统计，截止到 2006 年 12 月，全球已经有 1900 多万 Web 站点，包含 130 多万个 IP 地址使用 PHP。

作者结合多年来使用 PHP 开发网站应用程序和教学的经验，在查阅国内外大量 PHP 文献的基础上，以最新的 PHP 5 为编程工具，MySQL 4.1 为 Web 数据库，编写了此书。全书共 16 章，第 1～第 2 章介绍了 Web 体系结构、Internet 通信协议、动态 Web 工作原理以及 HTML 常用标记的使用。第 3 章介绍 JavaScript 客户端脚本语言的编程技术，熟悉这一章内容，有助于开发高质量的服务器端动态 Web 程序。第 4～第 5 章为 PHP 入门知识，介绍使用 PHP 的 Web 环境的构建、PHP 程序设计基础。第 6 章介绍 PHP 面向对象编程技术。第 7 章介绍 PHP 5 的常用内部函数。第 8～第 9 章介绍 MySQL 4.1 数据库系统以及如何使用 PHP 访问 MySQL 数据库的编程技术。第 10 章介绍 Session、Cookie 的使用和用户身份认证。第 11～第 15 章介绍 PHP 5 与 XML 的整合，包括 XML 文档的解析、Web 服务、XML 与数据库的数据交换、WDDX 数据交换技术。第 16 章介绍一个使用 PHP 开发的网络考试应用系统，通过这一综合实例，读者可以从中学会如何综合利用 PHP、JavaScript 和 SQL 命令，完成一个 Web 应用系统的开发过程。

本书内容丰富，结构编排合理，由浅入深地介绍了 PHP 5 动态网页编程的知识和应用，并通过大量的实例辅助说明，帮助读者掌握 PHP 5 的应用知识。本书编写中，注重培养读者的动态网页设计的能力。此外，本书每章后面附有习题和实验题，以供读者巩固每章所学的知识。

本书主要面向从事 Web 开发的初中级用户，可作为高等院校本科或专科计算机、电子商务、信息管理与信息系统以及相关专业的 Web 开发课程的教材或教学参考书，也可供 PHP 编程人员参考。

本书配有免费电子教案，可以从清华大学出版社网站（www.tup.com.cn）下载。

本书由徐辉主编，徐辉、卢守东、蒋曹清、陈绯、李菲编写。其中第 1 章、第 10 章、第 12～第 16 章由徐辉编写，第 2 章由李菲编写，第 3～第 4 章、第 7 章由蒋曹清编写，第 6 章、第 9 章、第 11 章由卢守东编写，第 5 章和第 8 章由陈绯编写，徐辉参与了第 5 章和第 8 章部分章节的编写工作。全书最后由徐辉统稿。

本书的出版得到清华大学出版社的大力支持，在此表示衷心的感谢。

由于作者水平有限，时间仓促，书中难免存在不足之处，敬请专家和读者提出宝贵意见，并将信息反馈给我们。

编　者

2007 年 12 月

目　录

第 1 章　Web 基 础

本章导读

Internet 的日益普及和广泛应用对人类的生活方式和工作方式产生了深刻的影响。人们通过 Internet 可以获取各种需要的信息，进行各种通信交流活动，例如，文献检索、发送电子邮件、网上购物、学术交流、电子广告、网上银行、网络会议、远程医疗、远程教学、网络电话等。Internet 让人们可以足不出户地在 Internet 这个浩瀚的信息海洋中遨游并得到无限的信息宝藏。特别是 Web 技术的出现，使越来越多的单位和个人能够在 Internet 上建立自己的网站。构建一个能提供这些信息的网站需要掌握各种不同的技术，包括网络技术、数据库技术、网页制作技术、网页编程技术、图形处理技术等。

本章是阅读全书的必备基础知识。首先介绍 Web 的体系结构及其组成要素，然后介绍 Internet 上常见的通信协议，包括 TCP/IP、HTTP、FTP 等协议，对统一资源定位地址（uniform resource location，URL）进行说明；介绍静态网页和动态网页的含义，实现动态网页的两种不同的 Web 工作模式。最后讨论以 XML 为基础的 Web 工作模式。

1.1　Web 概 述

Web 是 World Wide Web 的简称，又称为万维网、WWW 或 W3，它是在 Internet 上运行的遍及全球的多媒体信息系统。它是由欧洲粒子物理实验室（CERN）研制的一个基于超文本（hypertext）方式的信息检索服务工具。它将位于全世界 Internet 上不同地点的相关数据信息有机地编织在一起，这些信息有多种类型，包括文本、图形、图像、声音和视频等，为用户提供了一种十分有效的浏览、检索及查询信息的方式。用户可以根据关键字来搜索和显示信息，而这些信息又包含了与其他信息的链接，单击这些链接，Web 会根据所链接的地址从一个信息资源位置跳转到另一个信息资源位置，从而看到另一条信息。另外，WWW 还可以提供“传统的”Internet 服务，例如 Telnet、FTP、Gopher 和 Usenet News（Internet 的公告板服务）等。

1.1.1　Web 的体系结构

Web 由 Internet 上的计算机、信息资源和网络基础设施组成，这些计算机通过一定的网络通信方法互相连接起来，使得它们之间以及其中的内容能够方便地相互链接。这些互连的计算机中，有的用作服务器，其余的用作客户机。通常认为，服务器（server）是为别的计算机提供共享资源的计算机，客户机（client）是请求和使用服务器资源的计算机。在服务器上可以运行多个服务器程序。服务器程序是用于侦听客户机并对客户机的请求作出

响应的程序。例如，运行电子邮件程序并与电子邮件服务器相连，电子邮件服务器程序会通知连接的用户是否有新邮件到达，在同意接收后会把新邮件传送给用户的电子邮件程序。需要说明的是，同一台计算机既可用作服务器也可用作客户机，这对软件开发者来说，为开发网络应用程序和 Web 应用程序提供了方便的测试环境。

Web 系统的结构采用了客户机-服务器（client/server，C/S）的体系结构，如图 1.1 所示。Web 服务器是服务器端的计算机和运行在它上面的 Web 服务器软件的总和。其中，Web 服务器软件是昼夜不停地运行的程序，负责监听 Web 浏览器发送到服务器的 Web 页面请求，并提供相应的 Web 页面，通过 Internet 回传到客户端的浏览器。常用的 Web 服务器软件有 Apache 和微软的 Internet Information Server（IIS）。Apache 软件可以运行在各种不同的操作系统平台上，IIS 软件只能运行于 Windows 平台上。

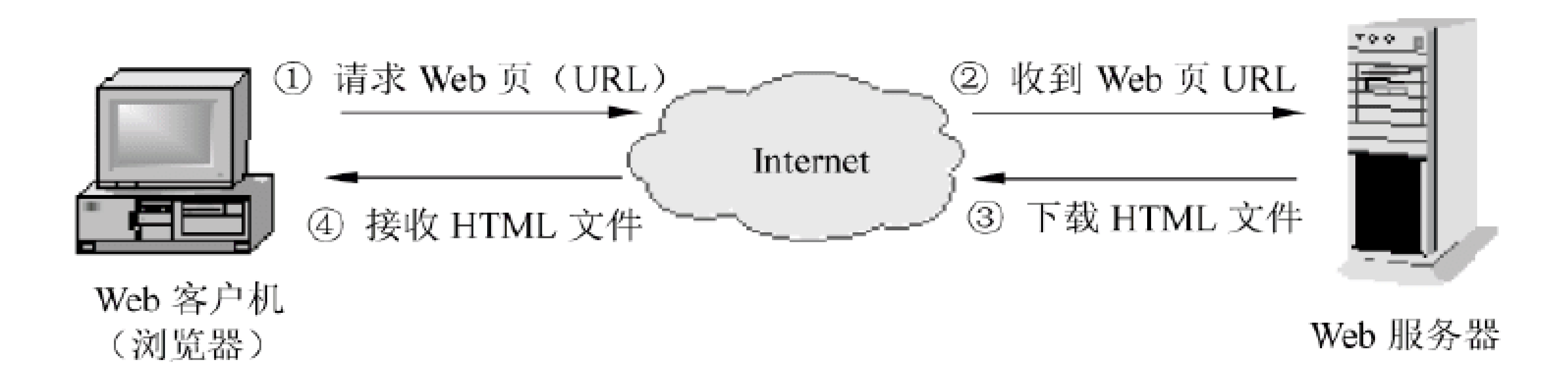


图 1.1　Web 的体系结构

Web 浏览器（browser）是用来解释 HTML 文档并完成相应转换和显示的程序。其主要功能是接收统一资源定位地址（URL）输入并发送 URL 请求，解释并显示由 Web 服务器传送回来的、由 HTML 写成的文档，包括嵌入在 HTML 文档中的 GIF 和 JPEG 格式的图像。此外，浏览器还可以根据用户的需要配置某些辅助应用程序，用来处理嵌入在 HTML 文档中的声音、视频、Flash 动画等多媒体信息。常用的浏览器是 Netscape 公司的 Navigator 和微软公司的 Internet Explorer（IE）。用户通过 Web 浏览器来访问 Internet 上的 Web 信息，把需要的信息从网上下载到本机，并在浏览器上显示。

Web 的信息资源以 Web 页面的形式存储在 Web 服务器上，Web 页面是用超文本标记语言（hypertext markup language，HTML）编写的文档，其文件扩展名通常是 htm 或 html。HTML 是一种文档布局的语言，用于定义 Web 页面的内容和外观。Web 页面中包括文字、图像、声音、动画、视频等各种多媒体信息，也允许包括用文本或多媒体表示的超链接，这样可以很方便地跳转到相关的其他 Web 页面中。这些 Web 页面构成一个网站的内容。因此，网站是存放在 Web 服务器上的一系列网页文档。

在基本的 Web 系统中，Web 服务器向浏览器提供服务的工作过程如下：

（1）用户启动浏览器程序，在浏览器中指定一个 URL，即通常所说的网址，它描述了信息所在的地址，浏览器便向该 URL 所指向的 Web 服务器发出请求。

（2）Web 服务器接收到浏览器的 URL 请求后，把 URL 转换成页面所在服务器上的文件路径名，查找相应的文件。

（3）若 URL 指向的是普通的 HTML 文档，Web 服务器直接将它传送给浏览器。如果 HTML 文档中含有用 Java、JavaScript、VBScript 或 ActiveX 等编写的小应用程序，服务器也将它们随 HTML 文档一起传送给浏览器。

（4）浏览器解释处理 HTML 文档，执行客户端程序，并按 HTML 格式显示页面内容。

1.1.2 Web 页面的功能

Web 页面可实现的功能主要表现在以下三个方面。

1. 主页功能

主页（home page）是进入一个网站首先看到的页面，提供到网站其他部分的链接，以此来指引用户。主页的合理设计非常重要，它反映客户的产品或服务的特点，是网站最重要的核心页面，是网站的灵魂所在。所以设计时要注意主页内容和样式的最佳平衡。一个主页上部有该网站名称、标志图片。名称下方有横向或左侧纵向的动态折叠菜单，而每一个动态折叠菜单下有多个子菜单，它们之间都有超链接，用户可根据需要浏览相关内容。

一般情况下，默认的主页文件名是 index.htm、index.html，而微软的 IIS Web 服务器软件则将默认的主页文件名定义为 default.htm。

2. 超链接功能

超文本是一种联机信息表示和管理技术，它把网页中的文本或图形与地理上分散存储的信息相互链接，这种相关信息的链接称为“超链接”。所谓超链接，就是一个网页文档中存在着指向其他相关文档的指针。通过一个文档中的超链接，实现从一个网页转到另一个相关的网页，从一个网站转到另一个网站。

3. 页面的交互性

交互是网站响应用户动作和选择的方式。交互式的网站会吸引更多用户的参与，允许用户选择自己想看的内容，而不是静态地显示信息。交互类型有交互式导航、交互式多媒体、交互式广告、用户之间的交互等。例如，用户之间的交互有论坛、聊天、视频会议等。

1.2 Internet 通信协议

Internet，亦称因特网或国际互联网，它是由各种不同类型和规模的、独立管理和运行的主机或计算机网络组成的一个全球性超级网络。Internet 使用的网络协议是 TCP/IP 协议，凡是连入 Internet 的计算机都必须安装和运行 TCP/IP 协议软件。

需要说明的是，因特网和万维网这两个概念并不相同。因特网是指全球公用计算机互联网络，这些计算机通过能进行相互通信的设备和物理线路连接起来。而万维网是因特网的组成部分，是一个巨大的文档集合，其中一些文档通过超链接来相互连接，可以用浏览器查看其内容，万维网的绝大部分文档是用 HTML 语言编写并驻留在世界各地的网站中。因特网在万维网还没有出现之前就已经在使用了，而且即使没有万维网，因特网仍然可以使用。

1.2.1 TCP/IP 协议

TCP/IP 协议是一个工业标准的网络通信协议集，是为广域网设计的，其中最重要的是 TCP 协议和 IP 协议。因此，通常将这诸多协议简称为 TCP/IP 协议。

TCP/IP 协议把整个网络分成四个层次：自上而下分别是应用层、传输层、网络层和物理链路层。它们都建立在硬件基础层之上。图 1.2 给出了 TCP/IP 参考模型和 OSI 参考模型的对照。

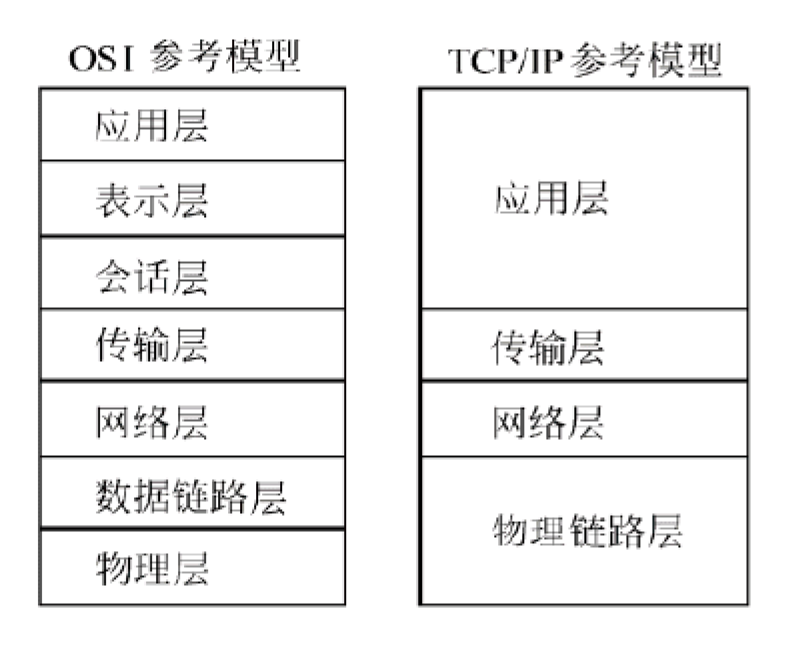


图 1.2 OSI 参考模型和 TCP/IP 参考模型的对照

1．应用层

应用层是 TCP/IP 参考模型的最高层，它向用户提供一些常用应用程序，如电子邮件服务、网页浏览服务等。应用层包括了所有的高层协议，并且不断有新的协议加入。应用层协议主要有：网络终端协议（Telnet），用于实现互联网中的远程登录功能；文件传输协议（FTP），用于实现互联网中交互式文件传输功能；简单邮件传输协议（SMTP），实现互联网中电子邮件的传递功能；网络文件系统（NFS），用于网络中不同主机间的文件系统共享；域名服务系统（DNS），用于实现网络设备域名到 IP 地址的映射服务；超文本传输协议（HTTP），用于在 Web 浏览器和服务器之间传输 Web 文档。

2．传输层

传输层又称 TCP 层，主要功能是负责应用进程之间的端到端通信服务，完成端到端的差错控制和流量控制，保证传输无差错，保持顺序，无丢失或无重复等。传输层定义了两种协议，即传输控制协议（TCP）和用户数据报协议（UDP）。

3．网络层

网络层又称 IP 层，负责处理互联网中计算机之间的通信，向传输层提供统一的数据包。IP 协议不检查某个数据分组是否真的到达目的地，这称为无确认传输；也不关心分组穿过互联网络的路径，甚至不确保各分组能否以其发送的顺序到达接收方，这又称为无连接协议。因此 IP 协议是无连接的、不可靠的。它的主要功能有三个方面，即处理来自传输层的分组发送请求；处理接收的数据包；处理互连的路径。

4．物理链路层

物理链路层包括物理层和数据链路层，是主机与网络的实际连接层。物理层是最底层。实际计算机的网络连接通常实现为网络接口卡及插入主设备的主板。主要功能是利用物理层传输介质为数据链路层提供物理连接，它是所有网络的基础。数据链路层是在物理层提供的位流传输服务的基础上，在通信的实体之间建立数据链路连接，传送以帧为单位的数据，采用差错控制、流量控制方法，使有差错的物理线路变成无差错的数据链路。

物理链路层的主要功能是接收网络层的 IP 数据报，通过网络向外发送；接收处理从网络上来的物理帧，抽出 IP 数据报，向网络层发送。

1.2.2　HTTP 协议

超文本传输协议（hypertext transfer protocol，HTTP）是专门为 Web 设计的一种网络协议，它属于 TCP/IP 参考模型中的应用层协议，位于 TCP/IP 协议族的顶层。因此，它在设计和使用中以 TCP/IP 协议集中的其他协议为基础。例如，它要通过 DNS 进行域名与 IP 地址的转换，要建立 TCP 链接才能进行文档传输。

HTTP 协议是 Web 服务器和浏览器之间的通信协议，负责传输 Web 文档。HTTP 是基于客户和服务器进行通信的基本模式，即请求和响应模式。浏览器和 Web 服务器之间的交互步骤如下：

（1）客户与服务器建立 TCP 链接；

（2）客户向服务器发出请求；

（3）服务器如果接受请求，就向客户发回响应信息，在响应信息中包括状态码和所要的文件（一般是 HTML 文档）；

（4）客户与服务器断开连接。

HTTP 协议的 Web 服务器的默认端口号是 80。

1.2.3　Telnet 协议

Telnet 是 TCP/IP 的一个应用层协议，即 Internet 远程登录协议,是 Internet 上强有力的功能。要使用 Telnet，在用户的计算机上必须安装和运行一个名为 Telnet 的程序。使用该功能，用户可以访问连入 Internet 的任何一台 Telnet 服务器，这时用户主机成为该远程主机的一个终端，并且可以访问各种所需的信息，或运行远程主机上的程序来求解各种复杂的问题。一切都是在远程主机上快速执行后再从远程主机返回服务的结果。

用户使用远程主机有两种情况：一种是要求用户有远程主机的账号和密码才能进行登录的；另一种是开放的，用户无须拥有自己的账号，即不用口令和用户名就能登录，这就是为公众开放的 Telnet 远程服务。

Telnet 协议的服务器端的默认端口号是 23。

1.2.4　FTP 协议

文件传输协议（file transfer protocol，FTP）是 TCP/IP 网络体系结构传输层的一种协议，是 Internet 上的一个重要应用。它的主要功能是把文件从一台计算机传递到另一台计算机，实现文件的上传和下载。通过 FTP 服务，允许 Internet 用户在两地之间双向传输文件，实现真正意义上的全球资源共享。FTP 与 Telnet 的不同之处是：前者可以实现两地间的文件复制（在首先取得对方用户名和密码的情况下），而后者是只能在取得远程主机权限（用户名和密码）的情况下让用户主机以终端方式共享远程主机的资源，而不能把远程主机上的文件复制到用户主机再传入用户自己的 PC 上。

FTP 是一种文件传输协议，体现的是一种服务。提供这种服务的机器叫做 FTP 服务器，又称 FTP 网站。FTP 服务器的默认端口号是 21。FTP 服务器有两类：一类是访问该服务器时必须拥有远程主机上的合法账户和密码，且拥有相应的访问权限；另一类是匿名 FTP 服务器，它允许没有账户和密码的用户仍然可以从远程主机获取文件，其目的是向公众提供

文件复制服务。在与这类匿名 FTP 服务器建立连接时，只在“用户名”栏输入 anonymous，在“密码”栏输入自己的 E-mail 地址作为口令即可完成连接过程。在 Internet 上有许多这样的、为公众开放的匿名 FTP 服务器，用户可免费下载程序代码、文件和软件等。

1.2.5 IP 地址

IP 地址是识别 Internet 网络中的主机及网络设备的唯一标识，以确定计算机在互联网上的位置。一个 IP 地址由 4 个字节组成，共 32 个二进制位。由 4 个用“.”分隔的十进制数组成，每个数不大于 255，如 210.36.139.6。

每个 IP 地址都由网络地址和主机地址两部分组成。网络地址代表一个网络段，每个网络段内的每台计算机都有唯一的主机地址。

IP 协议规定了 A～E 五类 IP 地址。其中 A、B、C 类是基本的，它们由 IP 地址的高位来区分，格式如图 1.3 所示。

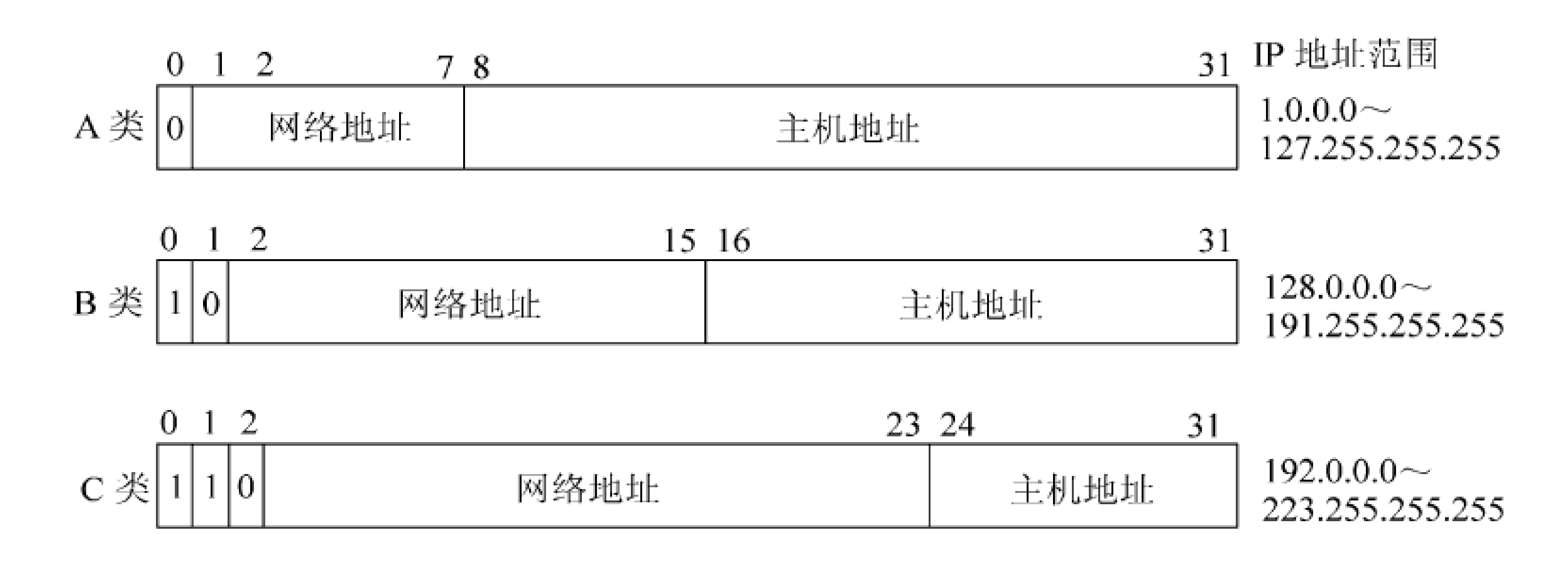


图 1.3 IP 地址格式

A 类 IP 地址的最高位为 0，接下来的 7 位为网络地址，其余 24 位为主机地址。A 类地址适用于在一个网络中主机数超过 65 534 台的场合，允许组成 126 个网络（0 和 127 被保留，其中 127.0.0.1 特指本机），每个 A 类网络内可以最多达 $2^{24}-2\approx1678$ 万台主机。

B 类 IP 地址的最高两位为 10，接下来 14 位为网络地址，其余 16 位为主机地址。它适用于一个主机数超过 254 台而小于 65 535 的网络，允许组成 2^{14}=16 384 个网络，每个网络内可包含 $2^{16}-2$=65 534 台主机。

C 类 IP 地址的最高三位为 110，接下来 21 位为网络地址，其余 8 位为主机地址。总共有 200 万多个 C 类地址，主要分配给小型网络使用。每一个 C 类网络的主机数最多为 254 台。

主机地址的末字节不能取 0 和 255 两个数。

1.2.6 域名服务器

IP 地址是联网计算机的地址标识，但是 IP 地址比较难记。所以 TCP/IP 协议中提供了域名服务系统（DNS），允许为主机分配域名。域名对人们来说具有一定的含义，容易记忆，例如，www.sina.com.cn。当用户在应用程序（如浏览器）中输入某一台主机的域名时，DNS 自动地将域名转换为与之对应的 IP 地址，然后把 IP 地址返回给应用程序，应用程序再利

用返回的 IP 地址与对应的主机连接。

需要说明的是：许多域名以 www 为前缀，也有一些域名没有 www 前缀，例如，msdn.microsoft.com。如果在浏览器地址栏中输入一个域名，如 sina.com.cn,浏览器首先按照输入的域名寻找 Web 站点，如果不能找到与输入域名一致的 Web 站点，浏览器会自动为域名加上 www 前缀，并寻找加上 www 前缀后的域名所代表的 Web 站点。

1.2.7 统一资源定位地址

统一资源定位地址（uniform resource location，URL）是为 Internet 的信息资源位置而设的一种编址方式，它指定 Internet 资源位于哪台计算机的哪个文件夹以及文件名。URL 的格式如下：

```
传输协议://主机 IP 地址或域名[:端口]/文件夹路径/文件名
```

例如，URL 为 http://java.csdn.net/n/20060728/93070.html。其中，http 为传输协议，java.csdn.net 为主机域名，n/20060728 为文件夹路径，93070.html 是要访问的网页文件。

URL 格式中各部分的含义如下所示。

（1）传输协议：传输协议指定了通信双方进行数据传输时采用的协议，常用的传输协议包括 HTTP、FTP、Gopher、Telnet、Mailto、News、Wais 和 File。HTTP 定位 Web 服务器上的文件，FTP 定位 FTP 服务器上的文件，Gopher 定位在 Gopher 服务器上的文件，Telnet 将用户连接到一个支持 Telnet 远程登录的服务器上，Mailto 指定电子邮件地址，News 定位在 Usenet 服务器上的文件，Wais 定位在 Wais 服务器上的文件，File 表示所访问的文件在用户的计算机或局域网上。如果在 URL 中没有指定通信协议，则 Web 浏览器默认使用 HTTP 协议。

（2）主机 IP 地址或域名：它指定存放资源的计算机，可以是计算机的 IP 地址，也可以是它的域名；端口号是可选项，指定服务器利用这个端口进行监听客户的请求。对于不同的协议，对应的服务器程序监听的端口号不同，都有各自的默认值，例如，Web 服务器默认监听 80 端口。只有当服务器改用其他端口号的时候，才必须在主机域名后附加相应的端口号。

（3）文件夹路径：文件夹路径由一系列文件夹名组成，用斜线（/）隔开，用于指定资源在服务器文件系统中的位置。如果没有指定文件夹路径，Web 服务器默认的起始位置是 Web 服务器的根目录，该目录是 Web 服务器为客户浏览器提供的所有可访问文件的起始点。

（4）文件名：文件名指定需要访问的资源。对于 HTTP 协议来说，文件名后缀决定 Web 服务器程序在响应请求时是运行程序还是下载一个文件。如果文件名后缀指定了一个程序，例如 php，那么 Web 服务器将运行该程序，生成 HTML 代码，最后把该 HTML 代码传送到用户的浏览器上并显示出页面。

当传输协议为 File 时，这种文件 URL 是对存储在本机硬盘或局域网上的文件的引用。例如，假设在 C 盘的\myweb 文件夹中存有一个名为 index.html 的网页文件，为了将它显示出来，可以输入如下的文件 URL：File:///c:\myweb\index.html。

需要注意的是：在文件 URL 中，File 关键字后面跟着三个斜线（/）。此外，对于微软的 IE 浏览器，可以省略“file:///”。例如，可以把 file:///c:\myweb\index.html 简写成 c:\myweb\index.html。

1.3　基于数据库的动态 Web 工作模式

1.3.1　静态网页和动态网页

1．静态网页

静态网页就是其内容由一些 HTML 代码组成的网页。这些 HTML 代码可以直接通过文本编辑器输入，并保存为.htm 或.html 文件。因此，静态网页的内容在创建时已经确定好了。任何用户在任何时候浏览静态网页都会看到相同的信息。图 1.4 显示了一个静态网页内容。

静态网页有其局限性。采用静态网页方式创建的网站只能简单地根据用户的请求传送现有的网页，无法实现用户参与的动态交互功能，不能显示用户个性化的特殊信息，更重要的是无法支持后台数据库。例如，要显示图 1.5 的个性化信息，其中姓名是用户提供的，时间是用户访问该页面时的时间。因此，其页面内容会根据用户提供的名字不同、访问时间的不同而改变，这仅仅依靠 HTML 是无法实现的。此外，HTML 也没有安全性，任何人都可以查看网页的 HTML 代码，没有能力防止其他人复制自己的 HTML 代码。因此静态网页有很大的局限性。

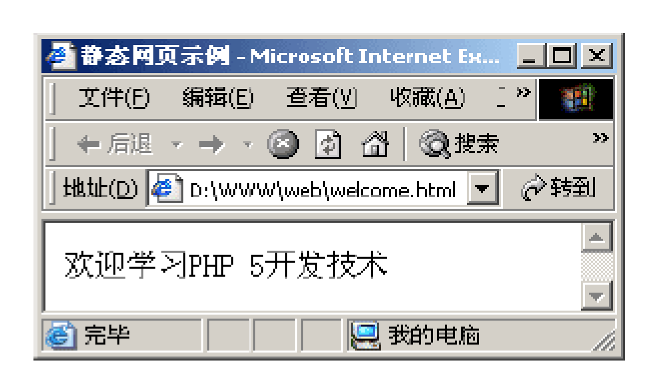


图 1.4　静态网页示例

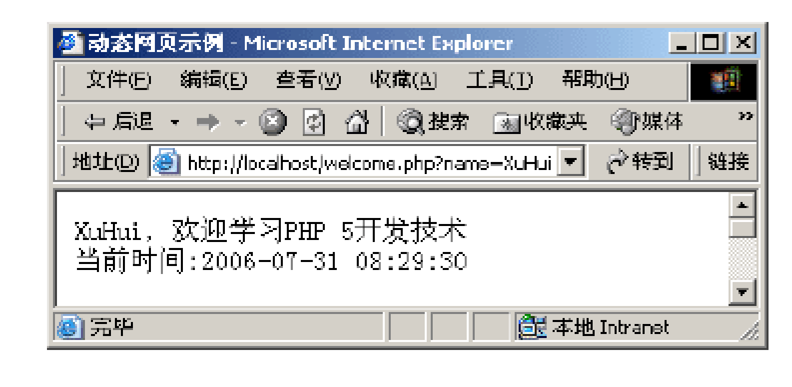


图 1.5　动态网页示例

2．动态网页

动态网页就是网页中加入程序或脚本，采用 ASP、PHP、CGI、ASP.NET、JSP 等技术动态生成的页面。图 1.5 和图 1.6 都是动态网页的例子。图 1.6 是访问 Google 网站，输入搜索关键字“Web 服务”后的搜索结果。输入不同的关键字，搜索结果也随之改变。



图 1.6　Google 搜索结果

动态网页的内容可以根据用户输入的数据或者其他数据源的数据，经过执行程序处理，把程序执行的结果动态生成 HTML 页面内容并传输到用户的浏览器，在浏览器上显示出来，从而实现客户端与服务器之间的交互。这也是静态网页与动态网页的最大区别。表 1.1 给出了静态网页和动态网页的比较。

表 1.1　静态网页和动态网页的比较

项目	静态网页	动态网页
内容	网页内容固定	网页内容动态生成
文件名后缀	htm、html 等	asp、php、jsp、aspx、cgi 等
优点	下载、浏览速度快，网页风格灵活多样	维护简单，修改方便，交互性好
缺点	交互性差，维护烦琐	占用系统资源
数据库	不支持	支持

基于数据库的动态 Web 网页是指需要从数据库获取部分或者全部内容而生成的动态网页。动态网页开发技术是指使网页成为动态网页的编程技术。互联网中的许多论坛、聊天室、留言本、电子商务网站等都是采用动态网页开发技术来实现的。这些技术可以访问常见的数据库，如 Access、SQL Server 2000、Oracle、MySQL 等。动态网页根据其程序执行位置的不同分为客户端动态网页和服务器端动态网页，从而动态网页开发技术分为客户端动态网页开发技术和服务器端动态网页开发技术。下面介绍这两种技术的工作原理。

1.3.2　动态 Web 的工作模式

1．客户端动态 Web 的工作模式

客户端动态网页是指在客户机的浏览器上执行程序，从远程数据库获取数据而动态生成的网页。当用户的 Web 浏览器访问 Web 服务器中的一个包含有客户端程序的页面时，需要从服务器下载编译过的客户端程序，并在用户的计算机上安装和运行，才能实现与 Internet 上的数据库服务器进行数据查询，生成动态的网页。图 1.7 是客户端动态 Web 的工作模式。

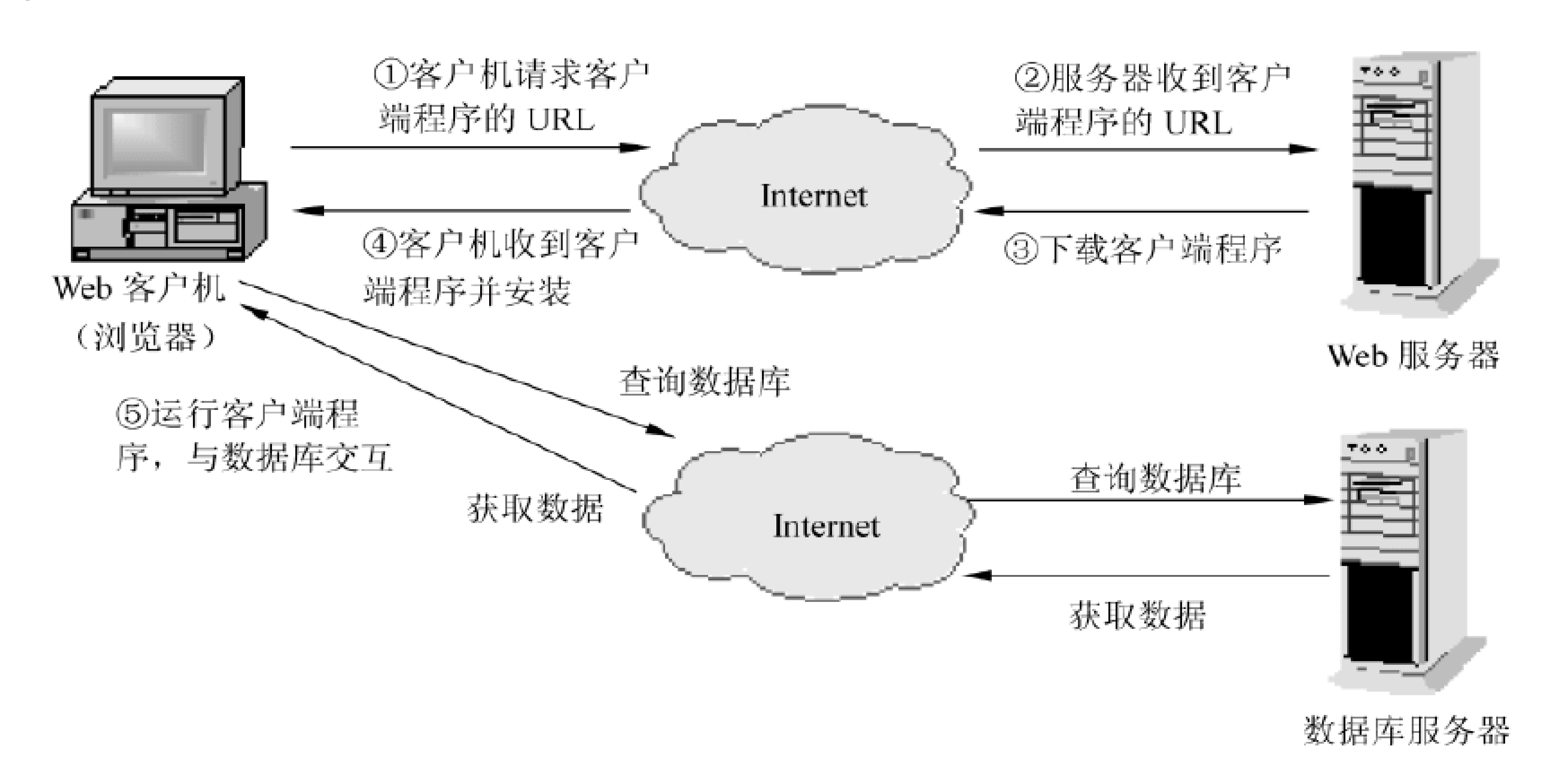


图 1.7　客户端动态 Web 的工作模式

目前，可以实现与服务器交互的客户端动态网页开发技术主要有 Java applet 程序和 ActiveX 控件两种。

1）Java applet 程序

Java 是一种用于开发基于 Internet 的应用程序的跨平台编程语言。利用 Java 创建的 Web 应用程序，称为 Java 小应用程序（Java applet 程序），这些 Java 小应用程序是用<applet>标记插入到 HTML 文档中，<applet>标记告诉浏览器从 Web 服务器下载 Java 小应用程序到浏览器，然后在浏览器上执行。由于 Java applet 程序是在浏览器提供的通用 Java 虚拟机（JVM）环境中运行，因此，它可以在任何操作系统、支持 Java 的浏览器上运行。出于安全考虑，Java applet 程序只能从 Web 服务器上接收数据，或者向 Web 服务器发送数据，不能对用户计算机中的文件进行任何读写操作。Java applet 程序通过 JDBC 提供的数据库支持，实现了客户端的浏览器与数据库服务器之间的数据交互。

2）ActiveX 控件

ActiveX 控件是微软推出的技术，可以使用许多不同的编程语言和环境来创建 ActiveX 程序，例如，C++、Visual Basic 和 Visual J++等，只要这些语言支持微软的 ActiveX 协议即可。由于 ActiveX 程序可以对用户计算机的操作系统和文件有完全访问权，因此，它存在一定的安全性问题，只有经过验证的、是公认信赖的，用户才可以下载、安装和执行 ActiveX 程序。ActiveX 程序通常用来创建内部网的应用。

当用户第一次访问含有 ActiveX 控件的网页时，会提示一个警告，将在计算机上安装一个程序，并说明了这个应用程序的来源。用户下载了 ActiveX 程序，并且安装、执行 ActiveX 程序，就可以不通过 Web 服务器而直接向数据库服务器发送数据或者从数据库获取数据。

需要说明的是，利用客户端脚本语言 JavaScript 或 VBScript 编写的客户端脚本程序，可以嵌入到 HTML 文档中，也可以保存为独立的文件，然后在 HTML 文档中通过<Script>标记来引用。这两种客户端脚本程序也是在浏览器上解释执行的，完成客户端数据验证、打开新的浏览器窗口、显示滚动文字、显示动态图片、处理用户在浏览器内触发的事件（如单击、键盘响应）等任务，但是它们与 ActiveX 控件、Java applet 程序不同，不能与 Web 服务器进行交互，因而，不是真正意义上的动态网页编程技术。只有包含了用户与服务器间进行交互并允许用户参与的内容的网页才算是真正的动态网页。

2．服务器端动态 Web 的工作模式

服务器端动态网页是通过在 Web 服务器上执行应用程序，从后台数据库中获取数据而动态生成的网页。最常用的服务器端动态网页技术是使用 HTML 窗体（表单），它是增强的 HTML 文档，是设计用户输入界面、收集输入的数据并发送给 Web 服务器的技术。当把 HTML 窗体数据提交给 Web 服务器时，在 Web 服务器上运行相关的应用程序，处理用户输入的数据，并动态地生成响应的 Web 页面，再发送到用户的浏览器并显示。Web 应用程序可以是编译过的可执行程序，也可以未经编译的脚本程序，还可以是两者的混合。图 1.8 是服务器端动态 Web 的工作模式。

常见的 Web 服务器端动态网页开发技术有 CGI、PHP、ASP、ASP.NET、JSP 等。下面分别对它们作简单的介绍。

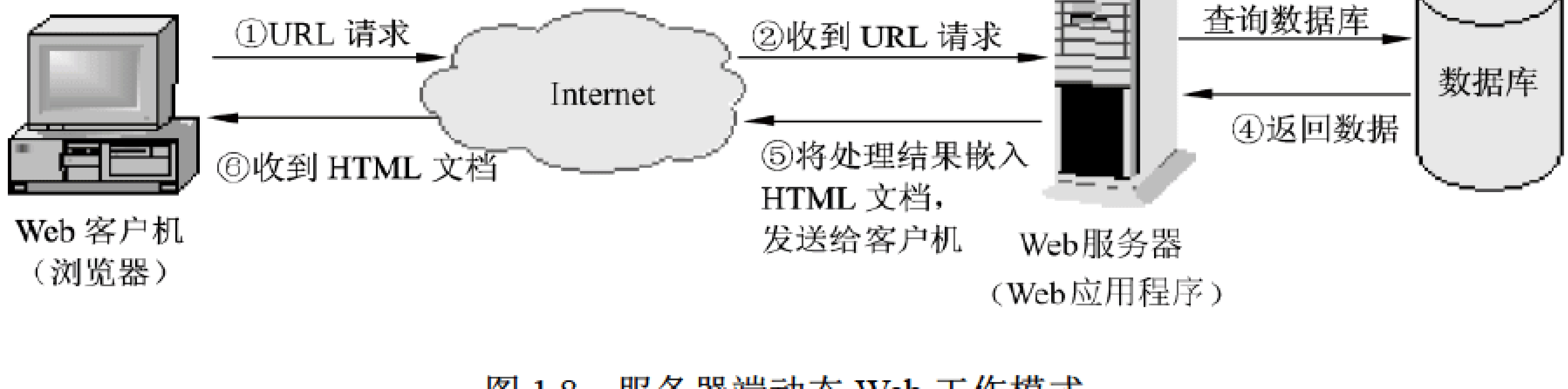


图 1.8　服务器端动态 Web 工作模式

1）CGI

公共网关接口（common gateway interface，CGI）是 Web 服务器与外部应用程序之间交换数据的标准接口软件，是最早的创建动态网页的机制。它可以用来创建动态 Web 应用程序，通过在 Web 服务器上运行 CGI 程序，输出一个动态的页面。CGI 是一种独立于语言的接口，CGI 程序可以使用任何能够访问环境变量和产生输出的编程语言来编写，如 C、C++、Perl、Shell 等。

CGI 与 Web 服务器的关系是：首先，Web 服务器必须支持 CGI 程序，并且 CGI 程序必须在 Web 服务器上运行。其次，客户端的 Web 浏览器常用 Post、Get 两种方式向 Web 服务器提交表单数据（图、表、文字的链接等），Web 服务器采用相应的数据传递方式向 CGI 应用程序传递数据。CGI 对数据处理后，将动态生成的 Web 页面发送给 Web 服务器，Web 服务器再把页面发送给请求数据的客户端。如果客户端用 Post 方式提交数据，Web 服务器按照标准方式向 CGI 输入和接收数据，CGI 同样按标准方式读取和输出数据。如果客户端用 Get 方式提交数据，在 UNIX 类系统中 Web 服务器通过环境变量方式把数据转交给 CGI 程序，CGI 程序必须从环境变量中读入数据，输出结果同样送到标准输出中。

虽然 CGI 程序能够生成动态的网页。但是，CGI 有以下缺点：

（1）学习难度大，使得开发 CGI 程序的门槛较高，相应的程序员就少了。

（2）CGI 占用较多的系统资源。因为用户请求访问 CGI 程序时，Web 服务器都运行 CGI 程序副本。例如，当四个不同用户的浏览器都请求访问 Web 服务器上的同一个 CGI 程序，那么 Web 服务器会同步地启动和执行该 CGI 程序四次。对于每个浏览器请求，Web 服务器都要启动一个新的 CGI 程序副本。这样在用户数很多的情况下，会造成 Web 服务器的内存严重不足，从而可能引起内存溢出。

为了克服 CGI 占用系统资源多的问题，Web 服务器开发商开发了专门处理 HTML 窗体输入数据的新技术，这样不必对每个浏览器的请求都打开一个程序的副本。这些与 Web 服务器通信的技术包括 Netscape 公司的网景公司服务系统应用程序编程接口（Netscape Server API，NSAPI）、Microsoft 公司的因特网服务系统应用程序编程接口（Internet Server API，ISAPI）。这些技术依赖于驻留在动态连接库（dynamic link library，DLL）中的过程。DLL 允许用存储在 DLL 中的一组过程为多个用户提供服务，而不用打开多个过程实例。

2）PHP

超文本预处理器（hypertext preprocessor，PHP）是一种易于学习和使用的、用来创建动态网页的服务器端脚本语言，是一种 HTML 内嵌式的语言（类似微软 IIS 上的 ASP）。

而 PHP 独特的语法混合了 C、Java、Perl 以及 PHP 式的新语法。它可以比 CGI 或者 Perl 更快速地执行。PHP 程序以解释方式来执行。

PHP 不断地更新并加入新的功能，几乎支持所有主流与非主流数据库；再以其高速的执行效率，使得 PHP 在 1999 年中的使用网站超过了 15 万。PHP 是开放源代码和跨平台的，使得 PHP 无论在 UNIX 或是 Windows 平台上都可以有更多新的功能。它提供丰富的函数，使得在程序设计方面有着更好的支持。

PHP 4 以上版本引入了 Zend Engine 引擎，极大提高了 PHP 程序的执行速度，满足更快的要求。经过最优化之后的效率，已经较传统 CGI 或者 ASP 等程序有更好的表现。而且还有更强的新功能、更丰富的函数库。

目前 PHP 最新版本是 5.1，PHP 是免费软件，可以从 PHP 官方网站 http://www.php.net 自由下载。有关 PHP 编程方面的内容从第 5 章开始学习。

3）ASP

ASP（active server pages）是微软推出的用以取代 CGI 的服务器端动态网页编程技术。其特点是：简单易学，功能强大；对客户端没有任何特殊的要求，只要有一个普通的浏览器就行。ASP 文件是在普通的 HTML 文件中嵌入 VBScript 或 JavaScript 脚本语言形成的。

ASP 程序的优点是：ASP 所使用的 VBScript 脚本语言直接来源于 VB 语言，秉承了 VB 简单易学的特点，非常容易掌握。把脚本语言直接嵌入 HTML 文档中，不需要编译和连接就可以直接解释运行。利用 ADO 组件轻松存取数据库。ASP 存取数据库非常容易，没有 CGI 那么难学。面向对象编程，可扩展 ActiveX Server 组件功能。不存在浏览器兼容的问题。可以隐藏程序代码。

ASP 程序的缺点是：解释执行，运行速度比较慢；只能在 Windows 98/2000/XP/2003 操作系统平台下运行。

4）ASP.NET

ASP.NET 的设计初衷是解决 ASP 程序开发“复杂”、“烦琐”等问题。ASP 虽然是在国内相当流行的动态网页开发技术之一，但是这种技术在被广泛使用的同时，也不断地暴露出问题（如脚本语言功能有限，应用处理逻辑与 HTML 标记混杂在一起从而不易分辨，性能不容易扩充等）。早在 1997 年，IIS 的开发人员就提出了 ASP.NET 的技术架构，而当时，ASP 面世刚满一年。

ASP.NET 彻底抛弃了脚本语言，而代之于编译式语言（如 VB、C#等），为开发者提供了有很大选择余地的、功能完善的编程语言。开发者可以选择.NET 框架支持的任何一种编程语言来创建更快、更可靠的 ASP.NET 应用程序。ASP.NET 允许用服务器端控件取代传统的 HTML 元素并充分支持事件驱动机制。也为开发者提供了强力的集成开发工具 Visual Studio.NET。目前最新版本是 Visual Studio.NET 2005。

5）JSP

JSP（java server pages）是允许用户将 HTML、XML 标记与 Java 程序组合动态生成网页的技术。JSP 用的编程语言是 Java。它是由 Sun 公司提出、多家公司合作建立的一种动态网页技术。该技术的目的是为了整合已经存在的 Java 编程环境（例如 Java Server 等），结果产生了一种全新的足以和 ASP 抗衡的网络程序语言。

JSP 的最大优点是开放的、跨平台的结构，运行速度比 ASP 快。它可以运行在几乎所

有的服务器系统上，包括 Windows NT、Windows 2000、UNIX、Linux、Windows 98 等。当然，需要安装 JSP 服务器引擎软件。Sun 公司提供了免费的 JDK、JSDK 和 JSWDK 软件供 Windows 和 Linux 系统使用。JSP 也是在服务器端运行的，对客户端浏览器要求很低。

1.4　基于 XML 的 Web 工作模式

上一节介绍的基于数据库的 Web 体系结构是客户机-服务器模式的体系结构。它一般是三层结构：客户端是一个浏览器，它将对页面的请求发送给 Web 服务器，显示返回的 HTML 文档；Web 服务器通过执行 CGI 程序或脚本程序来动态生成 HTML；后台数据库作为第三层。各层由专用的、固定的方法进行控制。这种体系结构存在着以下限制。

（1）用户被限制在客户端浏览器上，请求的结构是固定的。

（2）Web 应用程序与其他应用程序之间不能共享数据，因而多个 Web 服务器协作处理一个请求，显得比较复杂。例如，一家公司想以电子方式把购买订单发送给销售商，如果公司的购买订单的存储格式与销售商的格式不一致，就必须把购买信息转换成能够与销售商的程序兼容的格式或者重新手工输入，原因就是数据格式不兼容。

（3）所有的内容都以 HTML 的形式传递，这限制了 Web 客户端进行后期处理的能力。因为 HTML 有一个致命的缺点：它只适合于人与计算机的交流，不适合计算机与计算机的交流。通过 HTML 表现出来的文字、图形内容很容易被人理解，但却不利于计算机程序去理解。HTML 的另一个问题就是它的标记集合是固定的，用户不能根据自己的需要增加标记；而且各种浏览器的规格不尽相同，要想使用 HTML 做的网页能够被所有浏览器正常显示，只能够使用 W3C（万维网联盟）规定的标记来创建网页。

（4）客户端和服务器端必须同步，客户端所得到的信息必须始终与服务器端保持一致。

解决上述限制问题的一种方法就是把数据翻译成兼容各种应用程序的标准格式，即用可扩展标记语言（extensible markup language，XML）描述数据。W3C 对 XML 做了如下描述："XML 描述了一类被称为 XML 文档的数据对象，并部分描述了处理它们的计算机程序的行为。XML 是 SGML 的一个应用实例。从结构上说，XML 文档遵从 SGML 文档标准。"同 HTML 一样，XML 也是一种基于文本的标记语言，都是从标准通用标记语言（standard generalized markup language，SGML）发展而来的。XML 与 HTML 的不同在于：XML 可以根据用户要表现的文档，自由地定义标记来表现具有实际意义的文档内容，例如，可以定义<文档名称></文档名称>这样具有实际意义的标记。而且 XML 不像 HTML 那样具有固定的标记集合，它实际上是一种定义语言的语言，也就是说，使用 XML 的用户可以定义无穷的标记来描述文档中的任何数据元素，将文档的内容组织成丰富的、完整的信息体系。

总的来说，XML 具有卓越的性能，它具有四大特点：便于存储的数据格式、可扩展性、高度结构化以及方便的网络传输。这些特点为创建开放、高效、可扩展、个性化的 Web 应用提供了一个崭新的起点。

基于 XML 的 Web 体系结构如图 1.9 所示。客户端可以是浏览器，也可以是应用程序。服务器向客户端传送的内容可以是 XML 文档或者 HTML 文档。因此，定义和处理存储在 XML 文件的数据的方法有两种：一种是在服务器端处理 XML，另一种是在客户端处理 XML。

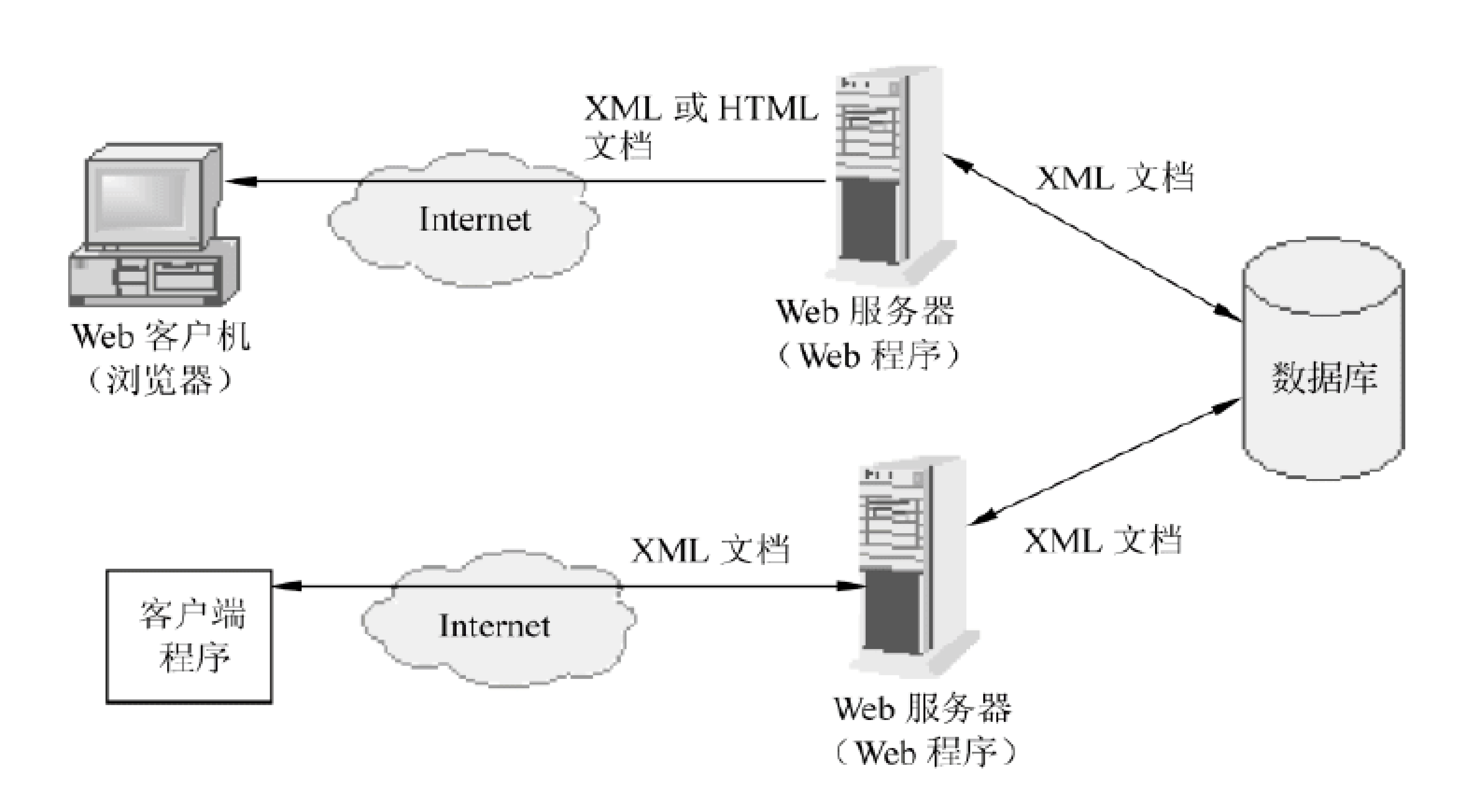


图 1.9　基于 XML 的 Web 体系结构

在服务器端处理 XML 的方法中，Web 服务器运行一个 Web 应用程序，从数据库中提取数据，把获取的数据转换成 XML 格式，然后用 XML 与 HTML 的转换程序把 XML 数据转换成 HTML 文件格式。这样 HTML 文件就可以通过网络传输到用户的浏览器。

在客户端处理 XML 的方法中，Web 服务器把数据库的数据转换成 XML 格式，然后通过网络把 XML 文件传送到客户端，由客户端的浏览器或程序对 XML 文档进一步处理。浏览器通过运行 XML 分析器（XML Parser），解释和翻译 XML 标记，使浏览器能够显示格式化的 Web 页面。目前，把 XML 文件转换成格式化 Web 页面的客户端技术包括层叠样式表（cascading style sheets，CSS）、可扩展的样式表转换语言（extensible stylesheet language transformations，XSLT）。其中，XSLT 既可用于浏览器也可用于服务器。

在基于 XML 的 Web 体系结构中，客户端和服务器是相对的。因为 XML 是 Web 服务器与 Web 服务器的应用程序之间进行数据交换的数据编码格式。接收 XML 数据的 Web 服务器扮演客户端角色，而发送 XML 数据的 Web 服务器扮演服务器的角色。在这种数据交换过程中，服务器可以将接收到的几个 XML 文档合并起来，或者将一个 XML 文档转化成另一种格式以满足客户端请求。这使得 XML 成为服务器与服务器应用程序之间交换数据的最佳媒介。这样，基于 XML 的 Web 体系结构就体现了 Web 应用的精髓——用简单的协议控制松散的、开放的资源集合。

实　验　1

1．在因特网中找出一个例子，当在浏览器中指定其域名或 IP 地址时，显示它的主页内容，然后抓取浏览器的主页界面图，保存到一个 Word 文件 Lab1_1.doc 中。

2．在因特网中找出一个 Word 文档类型的文件的 URL，在浏览器中显示该 Word 文档内容，然后抓取浏览器窗口图形，保存到一个 Word 文件 Lab1_2.doc 中。

3．在因特网中找出一个基于数据库的动态 Web 网页的 URL，在浏览器上显示它的页面内容，再抓取浏览器窗口图形，保存到 Word 文件 Lab1_3.doc 中。

抓图的简单方法是：首先，在屏幕上显示需要的内容时，按 Print Screen 键，将当前屏

幕图像复制到剪贴板中。如果按 Alt+Print Screen 组合键，则将当前活动窗口图像复制到剪贴板中。然后在 Word 文档窗口中，按 Ctrl+V 组合键，将剪贴板的图像粘贴到 Word 文档的当前位置。

习　题　1

1. 因特网和万维网之间有什么联系和区别？
2. 什么是客户机-服务器结构？
3. 什么是 URL？URL 由哪几个部分组成？
4. 本地主机（localhost）的 IP 地址是多少？请说明它有哪些用途？
5. 什么是主页？网页文件的扩展名有哪些？
6. 什么是 Web 服务器？它的主要作用是什么？
7. 什么是通信协议？
8. 什么情况下需要在 URL 中指定端口号？
9. 如何理解基于数据库的 Web 体系结构？它可分为哪两种模式？
10. 什么是客户端脚本程序？常见的客户端脚本语言有哪些？
11. 什么是服务器端脚本程序？常见的服务器端脚本语言有哪些？
12. 如何理解基于 XML 的 Web 体系结构？
13. TCP 协议和 IP 协议各有哪些主要作用？
14. 所有连接到因特网的计算机都采用什么协议？
15. 什么是超文本？
16. 在何处执行 CGI 程序？在何处执行 JavaScript 程序？在何处执行 Java applet 程序？
17. 什么是 HTTP？HTTP 协议的主要功能有哪些？
18. DNS 域名服务器的任务是什么？
19. 什么是静态网页？什么是动态网页？

第 2 章 HTML 基 础

本章导读

HTML 是 hypertext marked language 的简称，即超文本标记语言，它是一种用来制作超文本文档的简单标记语言。用 HTML 编写的超文本文档称为 HTML 文档，它独立于各种操作系统平台（如 UNIX、Windows 等）。自 1990 年以来 HTML 就一直被用作 World Wide Web 的信息表示语言，用于描述网页的格式布局设计及其与 WWW 上其他网页的联结信息。

本章是网页编程的基础知识。首先介绍 HTML 文档的句法结构及其基本标记的使用，接着介绍了超链接标记的使用方法，然后重点讲解了 HTML 中两个最重要的元素：HTML 表格和 HTML 表单。最后简单介绍了 Dreamweaver MX 2004 的使用。

2.1 HTML 文档结构

在万维网中传送的文档，绝大部分使用超文本标记语言编写，这些文档称为 HTML 文档。一般的 HTML 文档是由若干对象（objects）构成的，其中必须有一个基本 HTML 文档，其他对象可以是图像、声音、视频等各种与基本文档存在链接关系的文档，但在基本 HTML 文档中，只允许两种元素存在：一种是 HTML 标记，另一种则是普通文本。下面利用百度搜索引擎的主页来讲解 HTML 文档的句法结构（见图 2.1）。



图 2.1 百度搜索引擎的主页

【例 2.1】 百度主页的部分源代码。

相应的代码如下：

```
<html>
<head>
    <meta http-equiv=Content-Type content="text/html;
    charset=gb2312">
    <title>百度——全球最大中文搜索引擎</title>
    ⋮
</head>
<body>
    ⋮
</body>
</html>
```

通过本例可以看出，整个 HTML 文档是由各种标记和文本组成，一个简单的 HTML 文档的组成结构如下：

```
<html>           <! --超文本文件开始-->
    <head>       <! --首部元素开始-->
    <title> 网页的标题 </title>
    </head>
    <body>       <! --主题元素开始-->
        文档主体,正文部分
    </body>      <! --主题元素结束-->
</html>          <! --超文本文件结束-->
```

可见，一个完整的 HTML 文档通常由以下三部分组成。

1．HTML 标记

格式：

```
<html>
⋮
</html>
```

这是定义 HTML 文档开始与结束的标记，也是 HTML 文档中最先出现的标识，表明这个文件的内容是用 HTML 语言来实现的。它必须成对出现，分别表示 HTML 文件的起始和结束。在<html>容器标记中又包括以下<head>和<body>两个部分。

2．头部信息

格式：

```
<head>
⋮
</head>
```

这一部分用来说明文档标题以及该页面的其他信息，它构成 HTML 文档的开头部分，在此标记对之间可以使用<meta>、<title>等标记，这些标记都是描述 HTML 文档相关信息的标

记，标记对之间的内容是不会在浏览器的框内显示出来的，但是其内容应该尽量简短。

3．文本体

格式：

```
<body>
 ⋮
</body>
```

这部分说明 HTML 文件的主体内容，在浏览器的客户区中显示，这是页面开发者的编写 HTML 文档的主要部分。

对于 HTML 标记有一些约定，主要包括以下几点：

- 超文本标记是用一对尖括号“< >”括起来的文本串，例如，第一行的<html>。
- 超文本标记一般成对出现，用带“/”的标记结束，成对出现的超文本标记也称容器元素。
- 有些标记只有起始标记而没有结束标记（也称空元素）。
- 超文本标记可以忽略字母的大小写。
- 构成容器元素的一对标记可以写在不同行，标记属性的相对位置不受限制。
- 超文本标记的注解部分可以用<! -- … -->来表示。它可以放在源文件的任何位置，因为它对网页效果设置没有任何影响，但其目的是为了增强文件的可读性。

2.2 HTML 基本标记的使用

2.2.1 Head 容器的标记

<head>标记主要用来提供网页文件的整体信息。包括标题栏名称、文件的网址、所采用的文档编码等。<head>标记用来告知浏览器这是文件标题的开头，最后使用</head>标记告知浏览器这是文件标题的结束点。HTML 文档的 head 是一个容器元素，在<head>容器元素中允许出现以下元素。

1．title 元素

格式：

```
<title>  标题文字  </title>
```

title 元素包含文档的标题。它不显示在浏览器窗口中，只显示在浏览器标题栏中。在起始标记（<title>）和结束标记（</title>）间，可以放入简述文档内容的标题。如果没有 title 元素，浏览器的标题栏将显示网页的文件名。

2．link 元素

格式：

```
<link name="text" rel="text" href="URL">
```

link 元素在当前文档和另一文档之间建立链接关系。name 属性给链接起一个名字，rel 属性描述了链接的类型，href 属性指向相关的文档。

例如，下面的 link 元素表示链接一个外部 CSS 文件 default.css。

```
<link href="default.css" rel="stylesheet" type="text/css">
```

3．meta 元素

格式：

```
<meta name="text" content="text">
```

meta 元素通常用来为搜索引擎 robots 定义页面主题，或者是定义用户浏览器上的 cookie；它可以用于鉴别作者，设定页面格式，标注内容提要和关键字；还可以设置页面的自动刷新时间间隔（秒），等等。

下面介绍一些有关 meta 元素的例子。

1）设置搜索引擎关键字

允许全文检索的页面，为了使 Internet 上的搜索引擎能够有效检索，在频道的首页文件<head>元素的 meta 标记中应该加入 Keywords 和 Description 元标记。

定义网页的关键词：

```
<meta name="keywords" content="你网页的关键词">
```

定义网页的简述：

```
<meta name="description" content="你网页的简述">
```

2）设置网页文档编码

```
<meta http-equiv="Content-Type" content="text/html; charset=GB-2312">
```

可以通过 meta 元素为浏览器指定显示当前网页需用的文档编码。如上述语句指定了当前网页语言编码是简体中文。如果将其中的“charset=GB2312”替换成“BIG5”，则该页面所用的字符集就是繁体中文 Big5 码。

3）刷新页面

用 META 标记实现页面刷新的语法形式为：

```
<META http-equiv="refresh" content="second,URL=targetURL">
```

此 META 标记指示浏览按照 content 中指定的时间间隔自动地跳转到 URL 设定的网址。其中，second 参数表示时间间隔（秒数）；targetURL 参数为要跳转的目标地址，它可以是站点内的网页文件名，也可以是另一个网站的 URL。如果没有设置 URL 属性，则浏览器自动刷新该网页。

【例 2.2】 设计一个网页文件 ex2_2.html，经过 5 秒后浏览器自动转到网易网站 www.163.com，显示其主页内容。内容如下：

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=GB-2312">
<meta http-equiv="Refresh" content="5;URL=http://www.163.com">
<title>META 标记示例</title>
</head>
<body>
```

正在转向网易网…

```
</body>
</html>
```

4．base 元素

格式：

```
<base href="URL" target="text">
```

base 元素指定一个显式 URL 用于解析对于外部源的链接和引用，如图像和样式表。当用户使用相对 URL 请求文档时，超链接也会正确地执行。Target 指定文档中所有链接的默认窗口，该属性主要用在使用框架结构的网页中，使用框架结构，同一浏览器窗口中可以容纳多个网页，同时在若干不同的框架窗口中显示。

5．script 元素

script 元素在文档中放置一个脚本。这个元素可以在 HTML 文档的 HEAD 或 BODY 中出现任意多次。脚本可以在 script 元素中或外部文件中被定义。

2.2.2 Body 容器的标记

<body>标记是 HTML 文件最重要的部分，它是一个容器元素，包含在其中的内容将显示在浏览器的主窗口中。

<body>容器元素中可以包含表 2.1 的常用元素。

表 2.1 常用元素

<h#>text</h#>	定义标题。# 表示标题的层次（较小的数字标记较重要的标题），text 表示标题的文本
<p>	标记文档主体中两个段落之间的间隔
<img src= “URL”>	把图像插入到文档中，其中 src 属性给出图像地址
text	定义超文本链接属性，并将结果返回给用户浏览器
<hr>	放置一个横穿浏览器窗口的水平线
<address>text</address>	标志一个作为邮递地址或电子邮件地址的文本块
 	在文本中强制换行

在 HTML 中，需要使用<body>元素的属性对页面进行一个整体的规划和设置，如页面的背景颜色、背景图案、页面留白以及大部分文字的颜色，等等。例如，在百度搜索引擎的主页中，其<body>元素的定义如下：

```
< body text=#000000 vLink=#0000cc aLink=#ff6600 link=#0000cc
 bgColor=#ffffff onload=document.f.wd.focus()>
```

body 元素本身的属性可以分为 3 类，如表 2.2 所示。

表 2.2 body 元素的属性

背景属性	bgColor	背景色
	background	背景图案
文字属性	text	正文文字颜色
	link	链接文字颜色
	alink	活动链接文字颜色
	vlink	已访问链接文字颜色
边距属性	leftmargin	页面左侧的左边距
	topmargin	页面顶部的上边距

2.3 超链接标记

Web 最大的特点就是不同的文件和资源之间的相互链接，超文本中的链接是其最重要的特性之一，使用者可以从一个页面直接跳转到其他页面、图像或者服务器。HTML 用<a>来表示超链接，英文叫 anchor，一个链接的基本格式如下：

```
<a href="URL">  链接文字  </a>
```

其中，标签<a>表示一个链接的开始，表示链接的结束；<a>可以指向任何一个文件源：HTML 网页、图片、影视文件等。href 属性则表示这个链接文件的路径，通过单击“链接文字”可以到达指定的文件。链接分为本地链接、URL 链接和目录链接。在各种链接的各个要素中，资源地址是最重要的，一旦路径上出现差错，该资源就无法从用户端取得。

例如，链接到网易站点的首页，可以表示为：

```
<a href="http://www.163.com/index.html"> 网易首页</a>
```

下面介绍超链接标记的几个常用属性。

1．target 属性

使用 target 属性，可以在一个新窗口里打开链接文件。

```
<a href="http://www.163.com/index.html" target=_blank>
```

网易首页

2．title 属性

使用 title 属性，可以让鼠标在超链接上停留片刻后，显示该超链接的文字注释。

```
<a href="http://www.163.com/index.html" title="网易中文站点"> 网易首页 </a>
```

如果希望注释多行显示，可以使用
作为换行符。

3．name 属性

name 属性通常用于创建一个大文件的章节目录。每个章节都建立一个链接，放在文件的开始处，每个章节的开头都设置 Name 属性。当用户单击某个章节的链接时，这个章节的内容就显示在最上面。如果浏览器不能找到 Name 指定的部分，则显示文章开头，不报错。

使用 name 属性，可以跳转到同一个网页文件的指定位置。要使用 name 属性，需要设置两个<a>元素：一个是设定 name 的名称，第二个是设定一个 href 指向这个 name。

```
<a name="C1">第 1 章</a>
<a href="#C1">参见第 1 章</a>
```

4．链接到 E-mail 地址

在网站中，经常会看到“联系我们”的链接，一旦单击这个链接，就会触发客户端的邮件客户程序，比如 Outlook Express，然后显示一个新建 mail 的窗口。用<a>可以实现这

样的功能。例如：

```
<a href = "mailto:info@163.com">联系网易</a>
```

2.3.1 本地链接

对同一台机器上的不同文件进行的连接称为本地链接，它使用 UNIX 或 DOS 系统中文件路径的表示方法，采用绝对路径或相对路径来指示一个文件。

假如正在浏览的这一页的绝对路径是 c:\study\HTML 教程\link01.htm，而这一页是相对于当前目录即“HTML 教程”的相对路径。如果 link01.htm 是浏览 HTML 教程之外的一页，该文件路径要以两个圆点（..）来表示上一层目录../../internet/IP 地址。现在，把这几种路径的表示方法写入链接中。

1．以绝对路径表示

```
<a href="c:\study\HTML 教程\link01.htm">文件的链接</a>
```

2．以相对路径表示

```
<a href="link01.htm">文件的链接</a>
```

3．链接上一目录中的文件

```
<a href="../../Internet/IP 地址">IP 地址</a>
```

一般情况下，人们是不用绝对路径的，因为资源常常是放在网上供其他人浏览的，写成绝对路径，当把整个目录中的所有文件移植到服务器上时，带有 C:\的资源地址用户将无法访问到。所以最好写成相对路径，避免了重新修改文件资源路径的麻烦。

2.3.2 URL 链接

URL 是英文 uniform resource location（统一资源定位地址）的缩写，是专为标识 Internet 网上资源位置而设的一种编址方式，通过它可以使用多种通信协议与外界沟通来存取信息。URL 地址一般由三部分组成，例如：

```
http://www.163.com/index.html
协议名    主机名    路径及文件名
```

其中，http 指明所采用的传输协议为 HTTP（超文本传输协议），www.163.com 指明要访问的服务器的主机名，index.html 指明所访问的页面的文件名。因而，如果用户希望访问某台 WWW 服务器中的某个页面，只要在浏览器中输入该页面的 URL，便可以浏览到该页面。

Internet 常用的通信协议如表 2.3 所示。

表 2.3　常用通信协议

协议名	功能说明	协议名	功能说明
File	本地系统文件	FTP	ftp 服务器
HTTP	WWW 服务器	Telnet	基于 TELNET 的协议

续表

协议名	功能说明	协议名	功能说明
Mailto	电子邮件	Gopher	GOPHER 服务器
News	Usenet 新闻组	Wais	WAIS 服务器

2.3.3 目录链接

前面所谈的资源地址，只是单纯地指向一个文件，但是，如果要直接指到某一个文件的上部、下部或中央部分，以上方法却是无法做到的。然而，要这样做也并不是毫无办法。可以使用目录链接。

制作目录链接方法是：在各种链接的各个要素中，首先把某段落设置为链接位置，在这里，使用 name 属性，它可以跳转到一个文件的指定位置。格式是：

```
<a name="链接位置名称"> </a>
```

接下来，设定一个 href 指向此链接部分的文件，定义链接格式如下：

```
<a href ="文件名#链接位置名称"> 链接文字 </a>
```

但如果是在一个文件内跳转，文件名可以省略不写。例如：

```
<a name="C1">第 1 章</a>
<a href="#C1">参见第 1 章</a>
```

2.4 HTML 表格

2.4.1 表格定义标记

表格是 HTML 中最为复杂，同时又是应用最为广泛的元素之一。因为表格元素除了在网页中以表格形式组织和显示数据之外，还经常会用于网页的布局与设计。下面通过百度搜索引擎主页中的表格设计来讲解一下 HTML 的表格元素。

【例 2.3】 百度主页中表格设计的源代码。代码如下：

```
<table cellSpacing=0 cellPadding=0 width=600 border=0>
  <tbody >
  <tr valign=top>
    <td width=92></td>
    <td height=62>
      <form name=f action=http://www.baidu.com/s>
<input class=ff maxLength=100 size=35 name=wd>
<input type=hidden value=3 name=cl>
<input type=submit value=百度搜索>
<br>
</form>
</td>
```

```
<td width=92>
<A href="http://www.baidu.com/search/jiqiao.html">搜索帮助
</A>
<br>
<A href="http://www.baidu.com/gaoji/advanced.html">高级搜索
</A>
</td>
</tr>
</tbody >
</table >
```

该代码的设计视图如图 2.2 所示。



图 2.2　百度搜索引擎主页的表格设计

通过本例可以看出，表格中所有行和列以及单元格中的内容必须被包含在一对<table>和</table>标记中，其基本结构的一般形式如下：

```
<table>                          <!--定义表格-->
  <caption> ...  </caption>      <!--定义表格标题-->
  <tr>                           <!--定义表的一行-->
    ⋮
    <td>  ...   </td>            <!--定义一个单元格数据-->
      ⋮
  <tr>
    ⋮
</table>                         <!--定义表格结束-->
```

1. 表格的标题

表格标题的位置，可由 align 属性来设置，其位置分别为表格上方和表格下方。下面为表格标题位置设置格式。

设置标题位于表格上方：

```
<caption align=top> … </caption>
```

设置标题位于表格下方：

```
<caption align=bottom> … </caption>
```

2．<tr>元素

<tr>元素表示表格中的行标记，表格中的每一行都必须包含在一对<tr></tr>标记中。行标记的一般形式是：

```
<tr align="?" bgcolor="?"> … </tr>
```

在<tr>标记中有两个属性：

- align 指定该行中单元格的对齐方式，如左对齐、居中以及右对齐；
- bgcolor 指定该行的背景颜色。

3．<td>元素

单元格是表格的基本组成元素，一个 td 元素表示表格中的一个单元格，包含在<tr>元素内的多个<td>元素构成表格的一行。单元格的一般形式是：

```
<td width="?" heigh="?" align="?" valign="?" bgcolor="?" background="?"
rowspan="?" colspan="?"> ... </td>
```

在<td>标记中的属性如下：

- width　指定单元格的宽度；
- height　指定单元格的高度；
- align　指定单元格水平对齐方式；
- valign　指定单元格垂直对齐方式；
- bgcolor　指定单元格的背景颜色；
- background　指定单元格的背景图案；
- rowspan　指定单元格的行跨度；
- colspan　指定单元格的列跨度。

2.4.2　表格属性的设置

1．表格的大小

一般情况下，表格的总长度和总宽度是根据各行和各列的总和自动调整的，如果要直接固定表格的大小，可以使用下列方式。

```
<table width=n1 height=n2>
```

width 和 height 属性分别指定表格一个固定的宽度和长度，n1 和 n2 可以用像素来表示，也可以用百分比（与窗口相比的大小比例）来表示。

例如，定义一个长为 200 像素，宽为 100 像素的表格如下：

```
<table width="200" height="100">
```

而定义一个长为窗口宽度的 20%，宽为窗口高度的 10%的表格，则如下：

```
<table width=20% height=10%>
```

2．表格的边框

边框是用 border 属性来设定的，它表示表格的边框风格。将 border 设成不同的值，有不同的效果。在作为布局使用时往往取默认值“0”，即不显示表格的边框，设置为 1 时，表格边框显示成三维的状态。书写格式为：

```
<table border=n >
```

3．单元格间距

单元格与单元格之间的线为格间线，它的宽度可以使用<table>中的 cellspacing 属性加以调节。格式为：

```
<table cellspacing=#>  #表示要取用的像素值
```

还可以在<table>中设置 cellpadding 属性，用来规定内容与格线之间的宽度。格式为：

```
<table cellpadding=#>  #表示要取用的像素值
```

4．表格内文字的对齐方式

表格中数据的排列方式有两种，分别是左右排列和上下排列。左右排列是以 align 属性来设置，而上下排列则由 valign 属性来设置。左右排列的位置可分为居左（left）、居右（right）和居中（center）三种；而上下排列基本上比较常用的有上齐（top）、居中（middle）、下齐（bottom）和基线（baseline）四种。格式参照<tr>、<td>标记：

```
<tr align=#>    #为 left、center、right 之一
<td valign=#>   #为 top、middle、bottom、baseline 之一
```

2.5 HTML 表单

2.5.1 表单标记结构

HTML 表单（Form）是 HTML 的一个重要部分，主要用于采集和提交用户输入的信息。表单在 Web 网页中用来给访问者填写信息，从而能获得用户信息，使网页具有交互的功能。

HTML 表单标记的基本结构如下：

```
<form action="URL"  method=[get|post] >
  表单主体                          <! --一般用于定义表单的常用控件-->
</form>
```

<form>元素括起整个表单，并给出一些基本定义。表单仅占用 HTML 文档的部分空间；实际上，一个 HTML 文档可以包含几个独立的、完成不同功能的表单。action 属性指定接收和处理表单信息的脚本 URL，method 属性表示发送表单信息的方式。method 有两个值：

get 和 post。get 方式是将表单控件的 name/value 对信息经过编码之后，通过 URL 发送；而 post 方式则将表单的内容通过 HTTP 发送，在地址栏看不到表单的提交信息。一般情况下，如果只是为取得和显示数据，用 get 方式；一旦涉及数据的保存和更新，那么建议用 post 方式。

下面分别介绍 HTML 表单的几种常用控件的使用。

2.5.2 单行文本框和多行文本框

单行文本框允许用户输入一些简短的单行信息，比如用户姓名、密码等。格式如下：

```
<input type="text" name="文本框名">
```

【例 2.4】 单行文本框表单，如图 2.3 所示。

请输入你的姓名： 提交

图 2.3　单行文本框表单

示例代码如下：

```
<form action="http://www.aaa.cn/yourname.asp" method="get">
请输入你的姓名：
<input type="text" name="yourname">
<input type="submit" value="提交">
</form>
```

多行文本框（textarea）主要用于输入较长的多行文本信息。格式如下：

```
<textarea name="文本框名" cols ="宽度" rows = "行数">
</textarea>
```

其中，cols 表示 textarea 的宽度，rows 表示 textarea 的高度。

【例 2.5】 多行文本框表单，如图 2.4 所示。

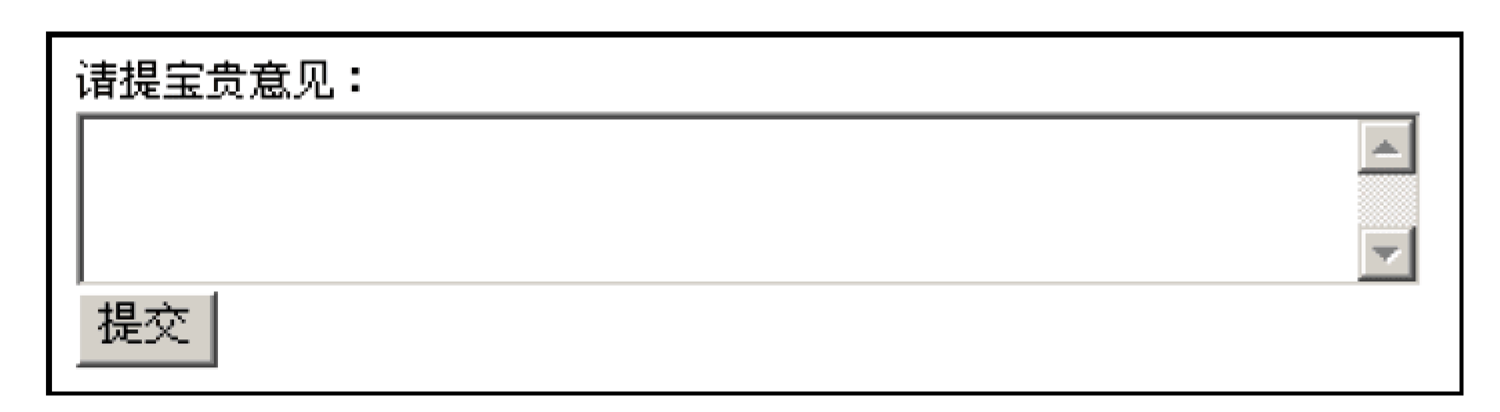


图 2.4　多行文本框表单

示例代码如下：

```
<form action="http://www.aaa.cn/suggest.asp" method="get">
```

```
请提宝贵意见：<br>
<textarea name="yoursuggest" cols ="50" rows = "3">
</textarea>
<br>
<input type="submit" value="提交">
</form>
```

2.5.3 命令按钮

命令按钮通常用于完成一定的操作，这由按钮的 type 属性值而确定。格式如下：

```
<input type="按钮类型" value="按钮标签文字">
```

type 属性指定按钮的类型，其值有三种：值为“Submit”，表示将表单的信息提交给表单的 action 所指向的文件来处理；值为“Reset”，表示清除表单的数据；值为“Button”，为一般按钮，不产生任何操作。value 属性值是显示在按钮上的文字。

2.5.4 单选按钮

使用单选按钮，让用户在一组选项里只能选择一个。格式如下：

```
<input type=" radio" name="按钮名">
```

【例 2.6】 单选按钮表单如图 2.5 所示，示例代码如下：

```
<form action="http://www.aaa.cn/choose.asp" method = "post">
<input type="radio" name="fruit" value = "Apple">苹果<br>
<input type="radio" name="fruit" value = "Orange">橘子<br>
<input type="radio" name="fruit" value = "Mango">芒果<br>
<input type="submit" value="提交">
</form>
```

请选择你最喜欢的水果：

○ 苹果
○ 橘子
○ 芒果
提交

图 2.5 单选按钮表单

2.5.5 复选框

复选框允许用户在一组选项里，选择一个或多个选项。格式如下：

```
<input type="checkbox" name="按钮名" [checked]>
```

其中，checked 是可选的，表示初始时选中复选框。

【例 2.7】 复选框表单，如图 2.6 所示。

请选择你喜欢的水果：

苹果
橘子
芒果
提交

图 2.6 复选框表单

示例代码如下：

```
<form action="http://www.aaa.cn/choose.asp" method ="post">
<input type="checkbox" name="fruit 1" value ="apple" >苹果<br>
<input type="checkbox" name="fruit 2" value ="orange">橘子<br>
<input type="checkbox" name="fruit 3" value ="mango">芒果<br>
<input type="submit" value="提交">
</form>
```

2.5.6 下拉列表框

下拉列表框既可以用做单选，也可以用做复选；如果要变成复选，加上 multiple 即可。用户用 Ctrl 键来实现多选。格式如下：

```
<select name="列表名" [multiple]>
   <option value="值 1">文本 1
   ⋮
   <option value="值 n">文本 n
</select>
```

你最喜欢的水果是： 苹果 提交
苹果
橘子
芒果

图 2.7 下拉列表框表单

【例 2.8】 下拉列表框表单，如图 2.7 所示。示例代码如下：

```
<form action="http://www.aaa.cn/choose.asp" method = "post">
```

你最喜欢的水果是：

```
<select name="fruit" >
<option value="apple">苹果
<option value="orange">橘子
<option value="mango">芒果
</select>
<input type="submit" value="提交">
</form>
```

2.5.7 隐藏域

隐藏域是指在表单上不显示任何内容，它允许表单使用 name 和 value 属性一起传送

预先设置好的信息。格式如下：

```
<input type=hidden name="名字" value="值">
```

2.6 Dreamweaver MX 2004 的使用

2.6.1 Dreamweaver MX 2004 简介

Macromedia Dreamweaver MX 2004（简称 DW MX 2004），是 Macromedia 最新开发的 HTML 编辑器，用于对 Web 站点、Web 页和 Web 应用程序进行设计、编码和开发。Dreamweaver MX 2004 包含有一个崭新、简洁、高效的界面，支持最新的 Web 技术，包含 HTML 检查、HTML 格式控制、HTML 格式化选项、HomeSite/BBEdit 捆绑、可视化网页设计、图像编辑、全局查找替换、全 FTP 功能、处理 Flash 和 Shockwave 等多媒体格式和动态 HTML、基于团队的 Web 创作。

Dreamweaver MX 2004 拥有如下新功能。

（1）简洁高效的设计和开发界面：界面更易于使用，可使用户的工作效率和工作质量均得到提高。

（2）“插入”栏的改进：简洁高效的新外观，占用更少的工作区空间。还新增加一个“收藏”类别，用户可以对“插入”栏进行自定义，将最常使用的对象放置在该栏上。表格编辑可视化：在表格中进行列调整操作时能看到实际效果。

（3）用户界面改进：可得到最大的可用工作区，更清晰地显示上下文和焦点，更易于使用和更具逻辑性。

（4）起始页：使用户能够访问最近使用过的文件，创建新文件和访问 DW MX 2004 资源。起始页会在用户启动 DW MX 2004 或尚未打开文档时显示。

（5）保存桌面选项：使用户可以选择当重新启动 DW MX 2004 时重新打开上一次使用的文档。

（6）完全支持 Unicode：DW MX 2004 支持 Internet Explorer 所支持的所有文本编码方式。用户可以使用几乎所有系统中安装的语言字体，DW MX 2004 会正确地显示和保存这些字体。

（7）安全 FTP：使用户能够完全加密所有文件传输。

（8）新式的页面布局和设计环境。

（9）增强的 CSS 功能：提供了一个更为精巧的方法来进行样式设计及提高设计交互性。

（10）动态跨浏览器验证：在用户保存文档时自动检查当前文档的跨浏览器兼容性问题。

2.6.2 Dreamweaver MX 2004 的站点管理

1．定制站点

在 Dreamweaver 中制作网站，必须定义一个本地站点，它是用户计算机上任意位置的一个文件夹，用于存放网站的所有文件。例如，在 E 盘根目录下创建一个名为 myweb 的文件夹，与网页相关的所有素材均保存在此文件夹下，一般需要将与网页相关的文件进

行分类，如图片文件放在 image 文件夹内，HTML 文件放在 html 文件夹内。

建好文件夹后，可按下列步骤建立本地站点：

（1）启动 Dreamweaver MX 2004，单击“站点”→“管理站点”命令，弹出“管理站点”对话框，如图 2.8 所示。

（2）单击“新建”按钮，新建一个站点，并为站点起名为“myweb”，如图 2.9 所示。



图 2.8 “管理站点”对话框

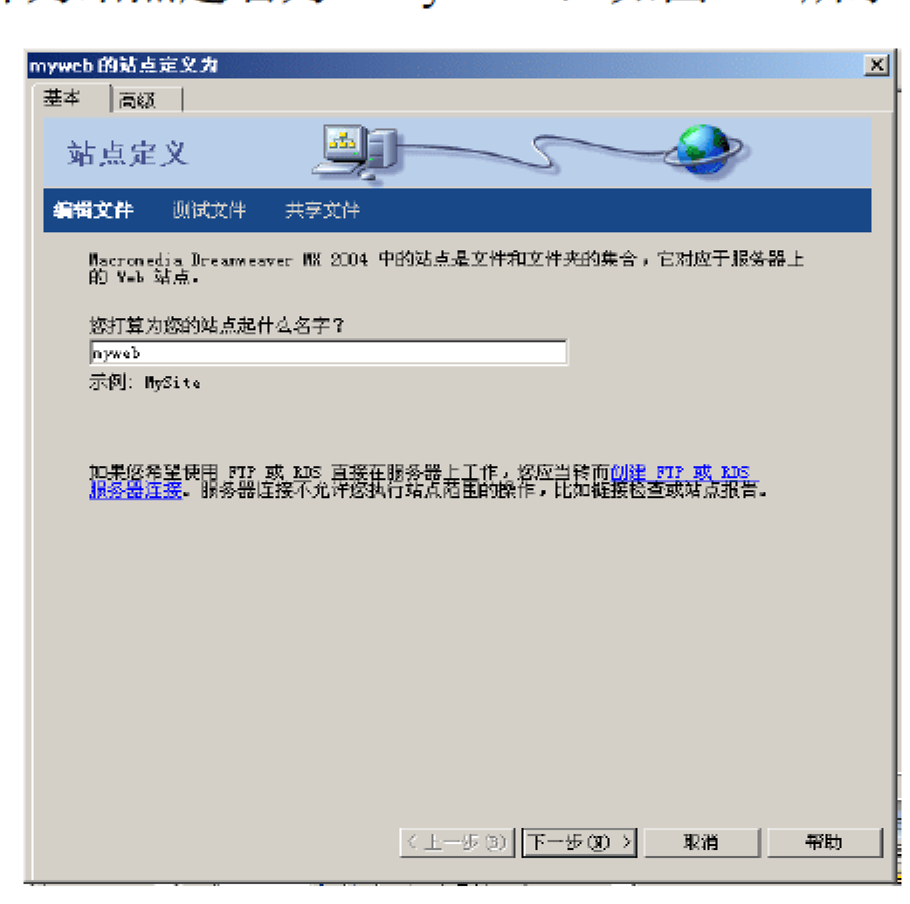


图 2.9 站点定义对话框

（3）单击“下一步”按钮，打开如图 2.10 所示的对话框。在该对话框中，选择是否使用服务器技术。此处选择“否，我不想使用服务器技术”单选按钮。

（4）单击“下一步”按钮，选择 E 盘根目录下的 myweb 文件夹作为站点的本地根文件夹，如图 2.11 所示。

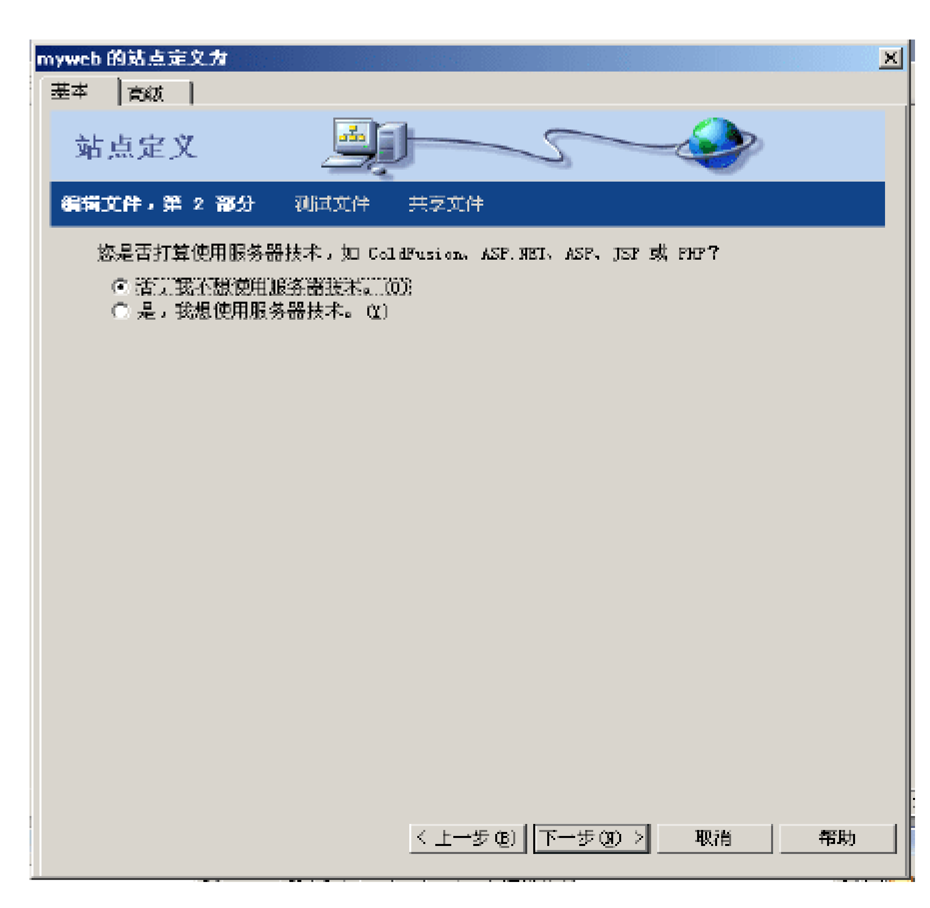


图 2.10 选择是否使用服务器技术



图 2.11 选择作为站点的本地根文件夹

（5）单击“下一步”按钮，选择将站点文件保存在服务器的位置，此时选择 E:\myweb，

如图 2.12 所示。

（6）单击“下一步”按钮，在该对话框中选择是否启用存回和取出文件。若用户在协作环境中工作，则可以从本地和远程服务器中存回和取出文件；若只有一个人在远程服务器上工作，则可以使用“上传”和“获取”命令，如图 2.13 所示。



图 2.12　站点文件保存在服务器的位置



图 2.13　选择是否启用存回和取出文件

（7）单击“下一步”按钮，打开如图 2.14 所示的对话框，在此显示了前面设置的站点信息。

（8）最后单击“完成”按钮，返回“管理站点”对话框，此时新建站点如图 2.15 所示。

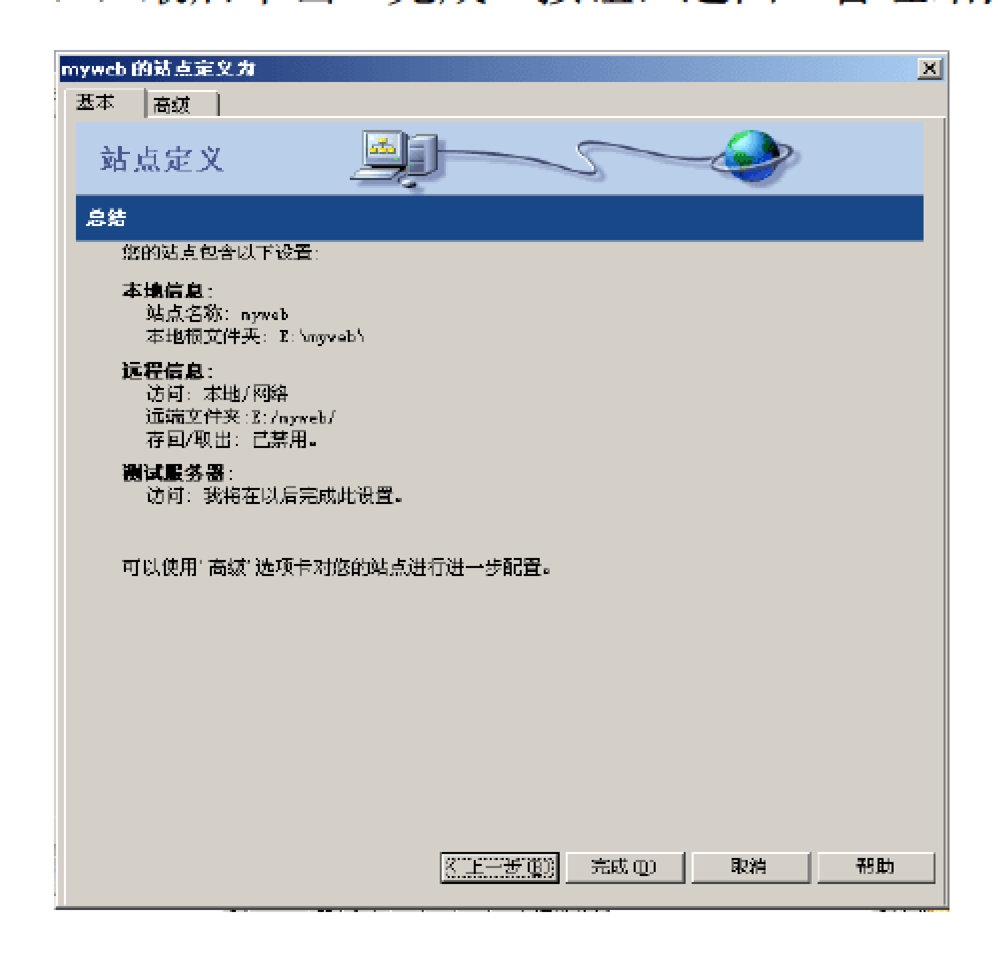


图 2.14　显示站点信息



图 2.15　“管理站点”对话框

2．站点管理

创建一个站点以后，可以随时对它进行编辑、复制、删除等操作。单击“站点”→“管理站点”命令，可以打开“管理站点”对话框，如图 2.15 所示。

编辑站点的操作如下：

打开“管理站点”对话框，选择要编辑的站点，此处为“myweb”，单击“编辑”按钮，打开“myweb 的站点定义为”对话框，选择“高级”选项卡。在此对话框中可以重新定义站点的名称，更改本地根文件夹的位置等，如图 2.16 所示。

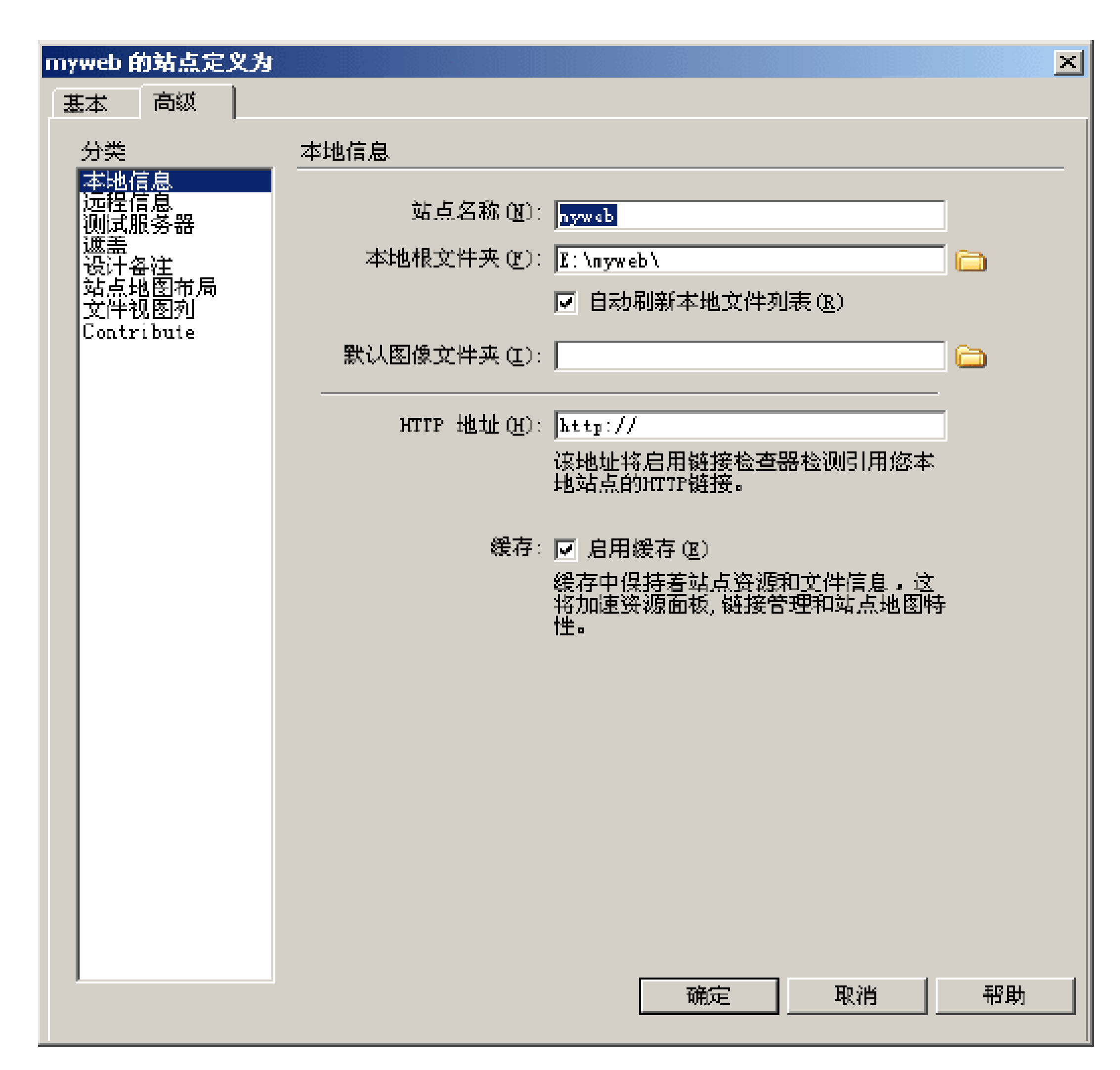


图 2.16 “myweb 的站点定义为”对话框的“高级”选项卡

2.6.3 网页文件的基本操作

1．创建网页

在 Dreamweaver MX 2004 主菜单中，单击“文件”→“新建”命令（快捷键为 Ctrl+N），弹出“新建文档”对话框，选定“常规”选项卡中“类别”下的“基本页面”选项，单击“创建”按钮即可新建一个页面，默认名称为 Untitled-1，如图 2.17 所示。

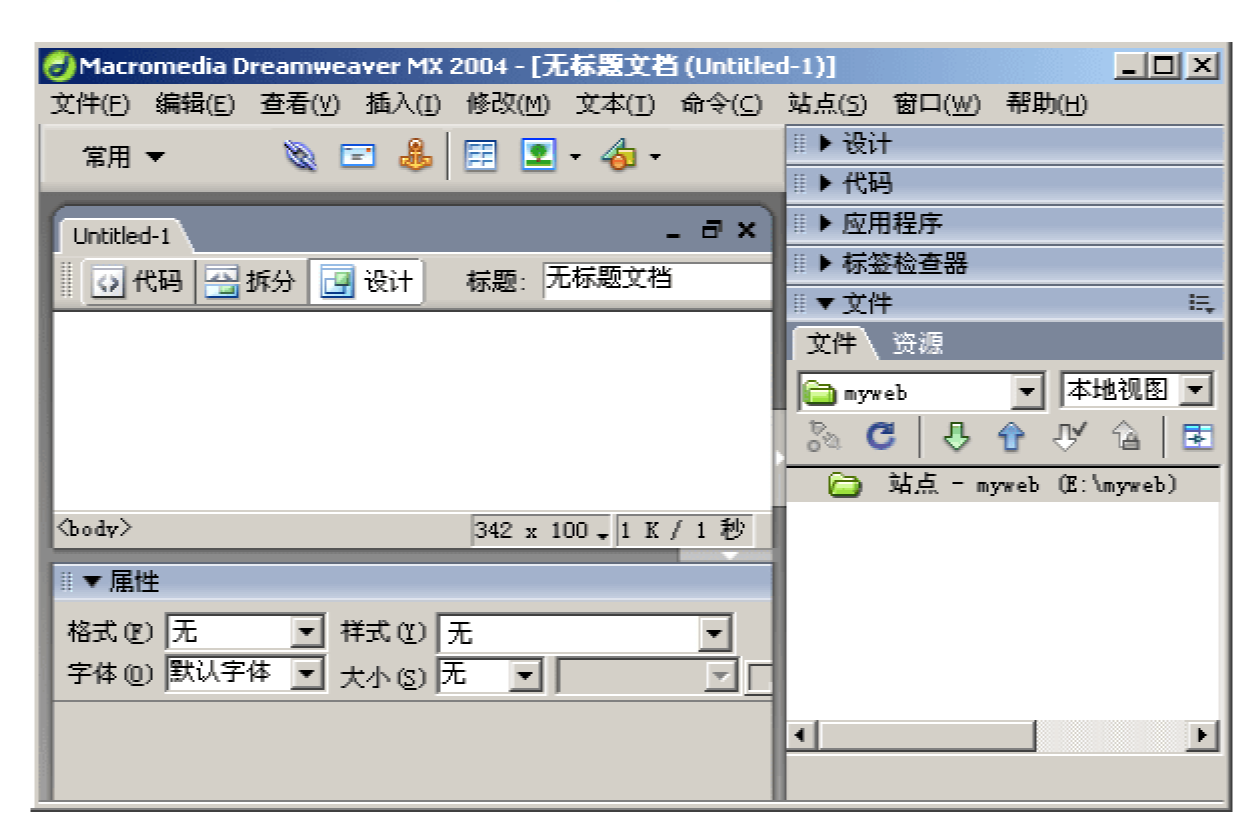


图 2.17 新建页面

2．打开网页

启动 Dreamweaver MX 2004，单击“文件”→“打开”命令（快捷键为 Ctrl+O），弹出“打开”对话框，选择要打开的文件，单击“打开”按钮即可打开该文件。默认情况下，系统打开 HTML 格式的文件。此外，还可以从“打开”对话框的“文件类型”中直接选择将要打开的不同格式的文件，例如，.js、.asp、.dwt、.as、.asr、.txt 等，如图 2.18 所示。

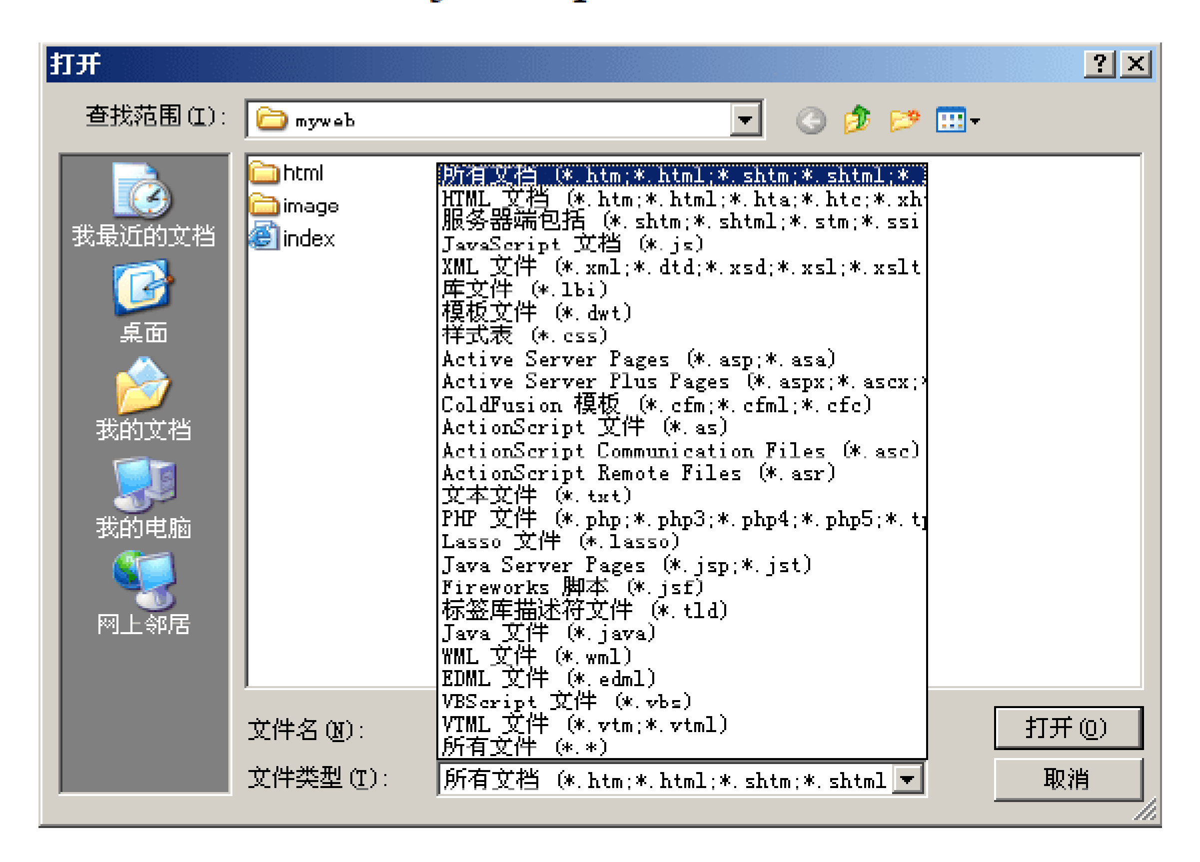


图 2.18 “打开”对话框

3．保存网页

在 Dreamweaver MX 2004 中，用户可以单击“文件”菜单进行保存文件、另存为文件，还可以将文件保存为模板。

将文件保存为模板文件以后，该文件将以模板的形式存在，如果其他文件应用该模板，则其格式会与之相同。值得注意的是，在第一次将文件保存为模板文件之前，首先要建立一个站点，如上一节建立的“myweb”站点，如图 2.19 所示。

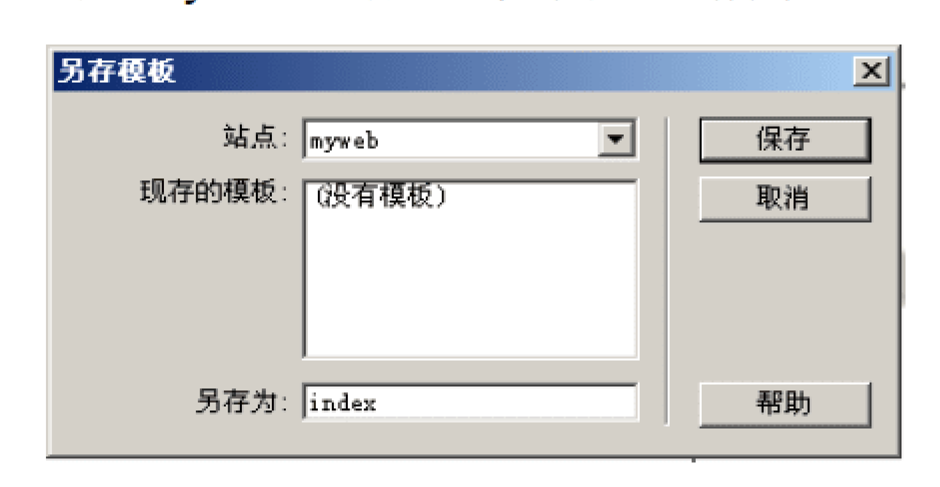


图 2.19 “另存模板”对话框

4．设置首页

每个网站都有自己的首页，首页是网站的统领，通过首页可以查看到该网站的所有信息。习惯上，一个网站的首页文件名通常设定为 index.htm 或 index.html，这样，当浏览网站时，Web 服务器便会自动找出 index.htm 文件，下载到用户的浏览器上显示。

首先单击“窗口”→“文件”命令，打开“文件”面板，然后在要设置成首页的网页上单击鼠标右键，打开快捷菜单，选择“设为首页”命令，如图 2.20 所示。

5．站点地图

利用站点地图，可以用图表的方式查看站点结构。如果没有设置网站的首页，则无法显示站点地图。设置首页之前，应该确定已存在网页，然后通过如下方法查看站点地图：单击“窗口”→“文件”命令，打开“文件”面板，然后单击面板上下拉列表中“地图视图”，站点地图如图 2.21 所示。

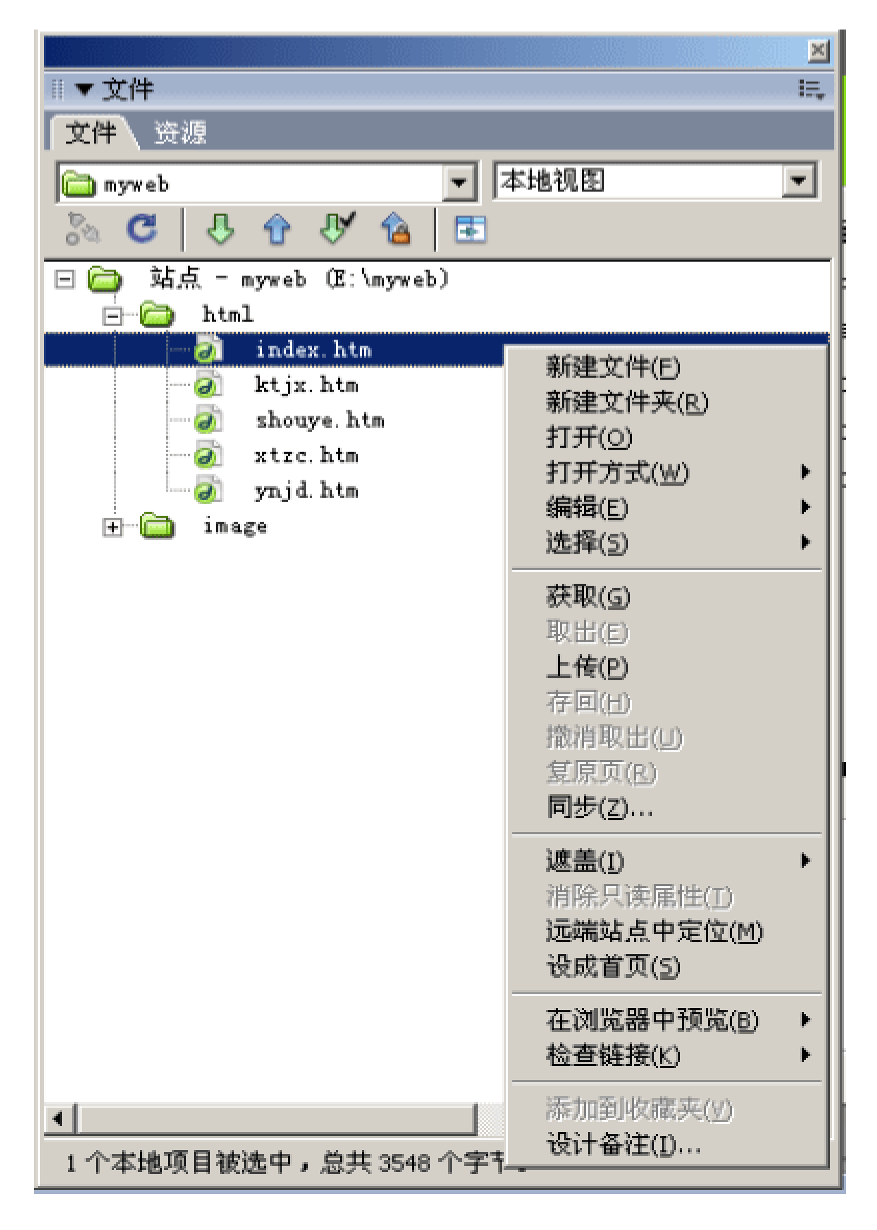


图 2.20　选择“设为首页”命令

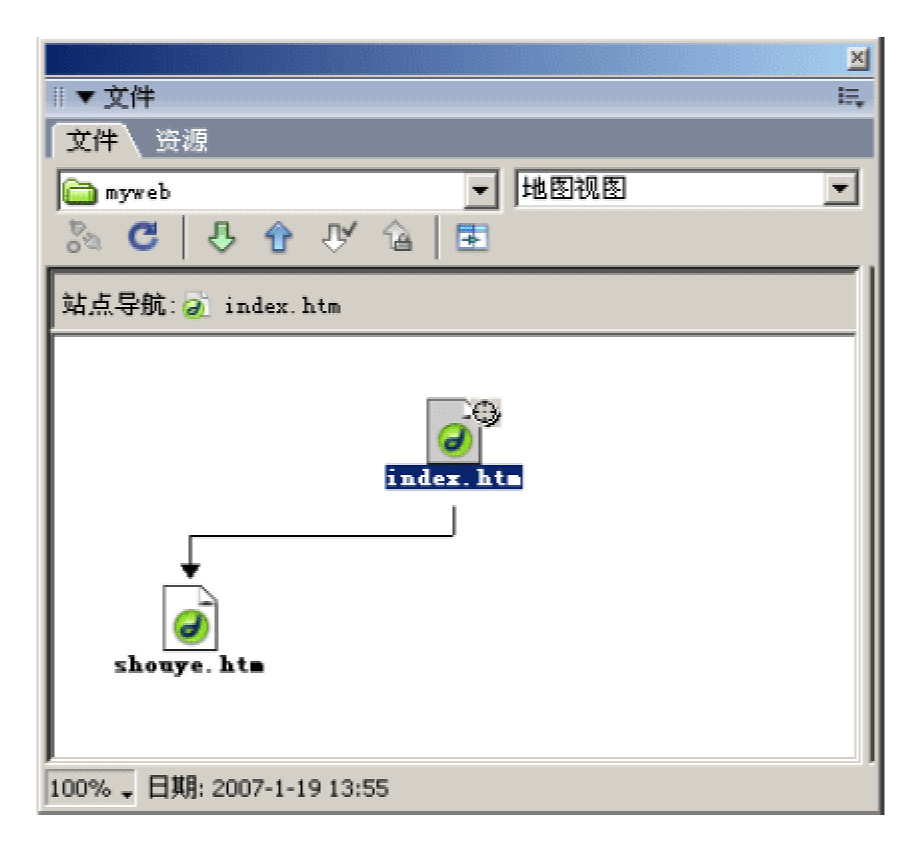


图 2.21　站点地图

实　验　2

1．创建一个如图 2.22 所示的 HTML 表单。

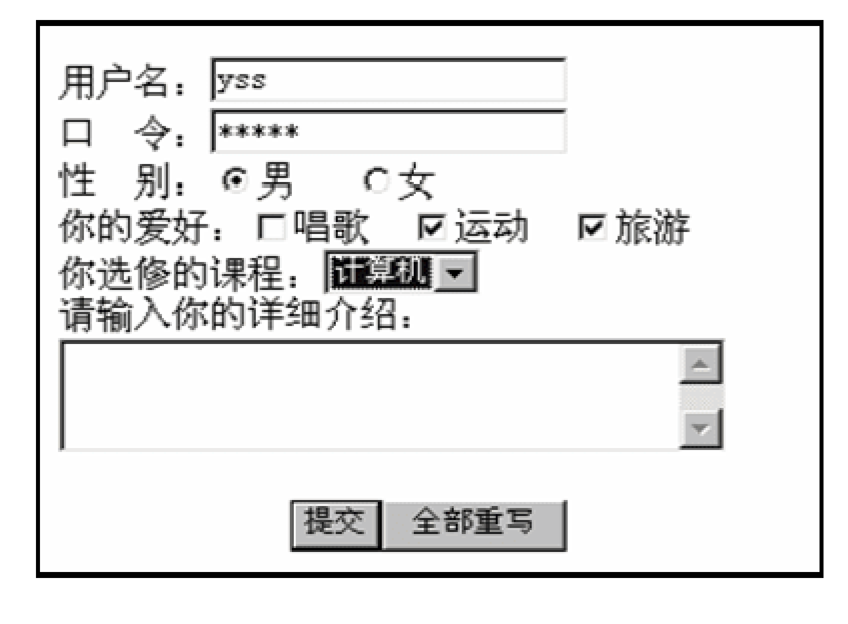


图 2.22　HTML 表单

2．在 D 盘建立一个 site 文件夹，建立一个名为 site 的站点并保存在该文件夹下。

3．在 site 站点中建立四个网页，将其中一个 index.htm 网页设置为首页，然后以“站点地图”的方式查看。

习　题　2

1．HTML 文档的组成结构由哪几部分构成？
2．HTML 文档的 head 容器元素通常包含哪几个元素？
3．HTML 文档的 body 容器元素通常包含哪几个元素？
4．Body 元素的属性有哪些？
5．超链接标记的基本格式及其常用属性是什么？
6．HTML 表格元素的基本结构是什么？
7．HTML 表格的属性设置一般有哪些？
8．HTML 表单的基本结构是什么？
9．HTML 表单的几种常用控件分别是什么？
10．Dreamweaver MX 2004 拥有哪些新功能？
11．如何定义 Dreamweaver 的本地站点？

第 3 章　JavaScript 客户端脚本语言

本章导读

JavaScript 是当前最为流行的一种客户端脚本语言。JavaScript 程序用于检测用户的活动，并对用户操作作出反应，比如，当光标掠过某个链接时，该链接变为另一种颜色，就是一种对用户操作的反应形式。JavaScript 程序为 Web 站点提供了导航帮助、滚动消息、对话框、动态图像、购物车等。通过 JavaScript 程序，可以控制 Web 页面的外观，也能够验证用户输入的有效性，而这些操作都无须与服务器进行交互，从而减轻服务器的工作负荷。JavaScript 程序也能够检测用户计算机上是否安装了某些插件，并在需要时能够从相应的站点上下载所需的插件。

JavaScript 是一种解释性的编程语言，它提供了变量、数据类型、条件语句、循环语句、函数、对象等一组基本的语言构件块，通过这些构件块，能够完成数学运算、操作日期和时间、分析字符串、操作数组和对象等。

本章首先从起源、发展和特点三个方面对 JavaScript 进行了概述。接着介绍了 JavaScript 的语言基础知识、对象模型、JavaScript 内置对象和浏览器对象的一些应用。

3.1　JavaScript 概述

随着 WWW 的迅速发展，采用 HTML 表示的静态信息资源，缺少动态的客户端与服务器端的交互，已经不能满足人们对其的需求，客观上需要一种可以提供动态交互的编程方法使之动态化。虽然可通过通用网关接口（common gateway interface，CGI）实现一定的交互，但由于该方法编程较为复杂，因而在一定程度上妨碍了 Internet 技术的发展。而 JavaScript 的出现，无疑为 Internet 网上用户带来了新的发展机会。可以这样说，JavaScript 的出现是时代的需求，是当今的信息时代造就了 JavaScript。

JavaScript 语言的前身叫做 Livescript。自从 Sun 公司推出著名的 Java 语言之后，Netscape 公司引进了 Sun 公司有关 Java 的程序概念，将原有的 Livescript 重新进行设计，并改名为 JavaScript。

JavaScript 的出现，使得信息和用户之间不仅只是一种显示和浏览的关系，而是实现了一种实时的、动态的、可交互式的表达能力。从而基于 CGI 静态的 HTML 页面将被可提供动态实时信息，并对客户操作进行反应的 Web 页面所取代。JavaScript 脚本正是满足这种需求而产生的语言。它深受广大用户的喜爱，是众多脚本语言中较为优秀的一种，它与 WWW 的结合有效地实现了网络计算和网络计算机的蓝图。因此，尽快掌握 JavaScript 脚本语言编程方法受到了广大用户的关注。

JavaScript 是一种基于对象和事件驱动并具有安全性能的脚本语言。使用它的目的是与 HTML 超文本标记语言、Java Applet 一起实现在一个 Web 页面中链接多个对象，与 Web 客户交互作用，从而可以开发客户端的应用程序等。它是通过嵌入或调入在标准的 HTML 语言中实现的。它的出现弥补了 HTML 语言的缺陷，是 Java 与 HTML 折中的选择，具有以下几个基本特点。

1. 基于对象的语言

面向对象程序设计力图将程序设计为一些可以完成不同功能的独立部分（即对象）的组合体。相同类型的对象作为一个类（class）被组合在一起，例如“公共汽车”对象属于“汽车”类。JavaScript 是一种基于对象的语言，虽然不是一种完全面向对象的语言，但提供了面向对象编程的重要特性，也就是说，它既能使用预先定义的对象,也能使用自己创建的对象。例如，在 JavaScript 中，不必创建“日期”这个对象，因为该语言已有这一对象，用户可以直接使用它。

2. 事件驱动的语言

当在 Web 主页中进行某种操作时，就产生了一个“事件”。事件几乎可以是任何事情，单击一个按钮、拖动鼠标等均可视为事件。JavaScript 是事件驱动的，当事件发生时，它可对之作出响应。具体如何响应某个事件取决于自己的事件响应处理程序。

3. 简单性

JavaScript 是一种描述性的脚本语言，不需要任何编译器，只要有一个字处理软件编写脚本程序，然后直接在 Web 浏览器中解释运行。它的基本结构形式与 C、C++、Java 十分类似，但它不像这些语言一样，需要先编译，而是在程序运行过程中被逐行地解释。它与 HTML 标识结合在一起，从而方便用户的使用操作。

4. 动态性

JavaScript 是动态的，它可以直接对用户输入作出响应，无须经过 Web 服务程序。它对用户的响应，是采用以事件驱动的方式进行的。当事件发生后，可能会引起相应的事件响应。

5. 跨平台性

JavaScript 依赖于浏览器本身，与操作环境无关，只要能运行浏览器的计算机，并支持 JavaScript 的浏览器就可以正确执行。不论使用 Macintosh 还是 Windows，或是 UNIX 版本的 Netscape Navigator，JavaScript 都可正常运行。

6. 安全性

JavaScript 被设计为通过浏览器实现信息浏览或动态交互，但它不允许访问本地的硬盘，并不能将数据存入到服务器上，不允许对网络文档进行修改和删除，从而有效地保证数据的安全。

7. 节省交互时间

随着 WWW 的迅速发展，有许多 WWW 服务器提供的服务要与浏览者进行交流，确认浏览的身份、需服务的内容等，这项工作通常由 CGI/PERL 编写相应的接口程序与用户进行交互来完成。很显然，通过网络与用户的交互过程一方面增大了网络的通信量，另一方面影响了服务器的服务性能。JavaScript 语言可以做到回应使用者的需求事件（如 form 的输入），而不用任何的网路来回传输资料，所以当一位使用者输入一项资料时，它不用经

过传给服务器处理，再传回来的过程，而直接可以被客户端的应用程序所处理。

JavaScript 是一种基于客户端技术的语言，用户在浏览中填表、验证的交互过程只是通过浏览器对调入 HTML 文档中的 JavaScript 源代码进行解释执行来完成的，即使是必须调用 CGI 的部分，浏览器只将用户输入验证后的信息提交给远程的服务器，大大减少了服务器的开销。

3.2　JavaScript 语言基础

JavaScript 同其他编程语言一样，也有它自身的数据类型、运算符、表达式等基本的语言特征。

3.2.1　数据类型

JavaScript 能够处理多种类型的数据。这些数据类型又可以分为基本数据类型和复合数据类型两类。基本数据类型是构造程序的最简单的元素；复合数据类型具有复杂的结构，后面章节介绍的对象就是一个典型的复合数据类型。

1. 基本数据类型

JavaScript 有五种基本数据类型。主要的类型有数值（number）、字符串（string）以及布尔型（boolean）类型。

（1）数值型：JavaScript 支持整数和浮点数。整数可以为正数、0 或者负数；浮点数可以包含小数点、也可以包含一个“e”（大小写均可，在科学记数法中表示“10 的幂”）或者同时包含这两项，如 1.35、1.25e3 就是浮点数。

（2）String 字符串类型：字符串是用单引号或双引号来说明的，如“The cow jumped over the moon.”。通常使用单引号来输入包含引号的字符串，如‘"a"<"c"’。

（3）布尔类型：可能的布尔值有 true 或 false，表示真或假。布尔值用于测试条件的真假。例如，先给布尔类型变量 bVar 赋值 true 或 false，然后可以在条件语句中测试条件是否成立：if（bVar）{完成某些操作}。

2. 复合数据类型

除了基本数据类型之外，JavaScript 还提供了一组更复杂的数据类型，称为复合数据类型。包括对象（object）、数组等。这里就数组进行简单介绍，而对象将留在后面介绍。

JavaScript 中数组是最常用的数据结构之一。可以把基本的数组结构看成单列的电子表格，列的每行包含不同的数据，并且每行都有标号。每行的标号严格按照数值顺序，第一行的标号为 0（程序员常常以 0 开始计数）。行号称为索引，为了存取某个数组元素，需要数组名和这个元素的行索引。因为索引值从 0 开始，数值元素的个数总是比数组的最大索引值大 1。

JavaScript 数组中的数组元素可以是任何数据类型，包括对象。同其他一些编程语言不同的是同一个 JavaScript 数组的不同元素可以包含不同的数据类型。比如名单、URL 以及颜色都可以存储在数组中。

与其他变量一样，使用数组前必须先声明数组，也就是创建数组对象。声明数组的语法有三种形式。

```
var myemptyarray = new Array( );
var myundefinedarray = new Array(n);
var myspecifiedarray = new Array(e0,e1,…,em);
```

第一种声明形式声明了一个空数组，它的元素个数为 0；第二种声明形式声明了一个有 n 个元素的数组，但每一个元素的值尚未定义；第三种声明形式声明了一个有 m+1 个元素的数组，它的各个元素的值依此为 $e_0,e_1,\cdots,e_m$。下面是使用这三种语法创建数组的示例。

```
Var array_name = new Array( );
```

创建了一个名为 array_name 的空数组。

```
Var array_name = new Array(100);
```

创建了一个名为 array_name、可以存放 100 个元素的数组。

```
Var array_name = new Array("red","blue","green",1,2,3);
```

创建了一个名为 array_name、有 6 个元素的数组，各元素的值依此是“red”，“blue”，“green”，1, 2, 3。

在声明数组时，无论是否指定了数组元素的个数，都可以根据需要调整元素的个数。JavaScript 实现按需分配内存，动态扩展数组。

下面通过一个例子来看看如何创建和使用一个数组，如何在 HTML 文档中嵌入 JavaScript 脚本。

【例 3.1】 创建和使用数组的范例，文件名为 ex3_1.htm。

```
<HTML>
<HEAD>
<TITLE> 编写 JavaScript 脚本 </TITLE>
<SCRIPT Language ="JavaScript">
var week = new Array(7);    //创建数组,共有 7 个单元,下标为 0~6
week[1]="Monday";           //将数组的 1 号单元赋值为"Monday"
document.write("today is "+week[1]);    //输出数组的 1 号单元的值
</SCRIPT>
</HEAD>
</HTML>
```

分析：

（1）var week=new Array（7）语句创建了名为 week 的一个数组变量，并且定义了数组的大小为 7，即创建了可以连续存放 7 个值的空单元。

（2）week[1]=“Monday”语句是给数组中的第 1 号单元赋值为 Monday。存储分配情况如图 3.1 所示。

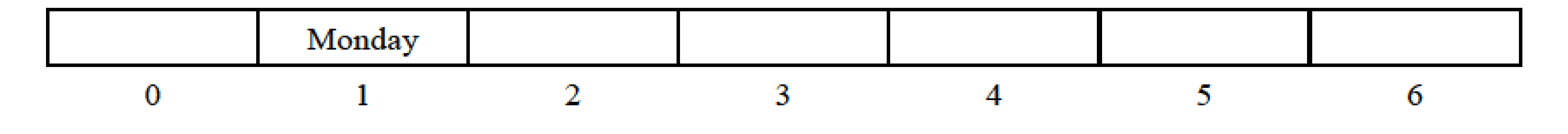


图 3.1　数组存储分配示意图

（3） document.write()是输出数组 week 第二个元素的值。

（4）每当在 HTML 文档中包含 JavaScript 脚本时，必须使用附有<SCRIPT>…</SCRIPT>标记对的命令行。这些标记提示浏览器程序解释标记中的所有文本。因为其他脚本语言（如 VBScript）也使用这些脚本标记，所以就必须用 Language ="JavaScript"明确标记所用脚本语言的确切名称是 JavaScript。因此，当浏览器知道脚本所用语言是 JavaScript 时，就使用内置的 JavaScript 解析器来处理这些代码，运行结果如图 3.2 所示。

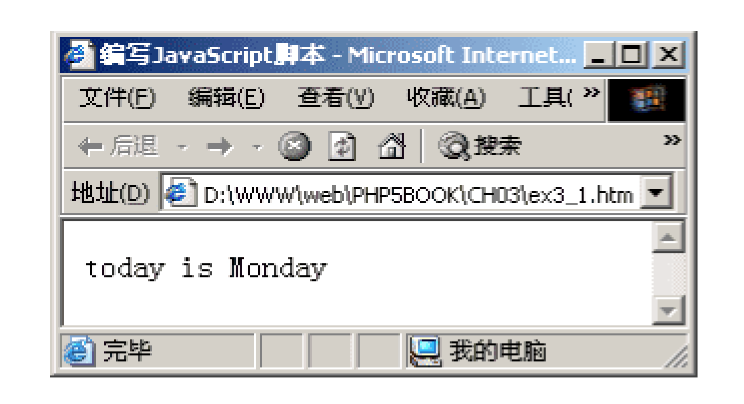


图 3.2　数组声明和使用示意图

3.2.2　常量和变量

1．常量

常量是指在程序执行过程中，其值不发生变化的量。通常分为数值型常量、布尔常量、字符型常量等。

（1）数值型常量　数值型常量可以使用十六进制、八进制和十进制来表示。其中，十进制数据的表示与日常所用的表示方法相同，对于较大的数值可以使用科学记数法表示，例如，23、0.58、–20、2.4E6（即 2.4×10^6）。

八进制数据的表示则需要在数据的前面添加一个 0，例如，034，0258。

十六进制数据的表示则需要在数据的前面添加一个 0x，例如，0x3E5。

（2）布尔常量　布尔常量只有 True 或 False 两种状态。它主要用来说明或代表一种状态或标志，以说明操作流程。它与 C、C++是不一样的，C、C++可以用 1 或 0 表示其状态，而 JavaScript 只能用 True 或 False 表示其状态。

（3）字符型常量　指使用单引号（“'”）或双引号（“"”）括起来的一个或几个字符。例如，“This is a book of JavaScript”、“3245”、“ewrt234234”等。

（4）特殊字符　null 是一个特殊的值，表示什么也没有。如试图引用没有定义的变量，则返回一个 null 值。除此之外，有些以反斜线（\）开头的不可显示的特殊字符，通常称为控制字符。例如，\b 表示退格，\n 表示换行，\r 表示回车符，\\表示反斜线。

2．变量

变量是用来存取数据、信息的容器。在 JavaScript 中变量用来存放脚本中的值，这样在需要用这个值的地方就可以用变量来代表，一个变量可以是一个数值、文本或其他一些东西。对于变量必须明确变量的命名、变量的类型、变量的声明及其变量的作用域。

（1）变量声明　JavaScript 是一种对数据类型变量要求不太严格的语言，所以不必声明每一个变量的类型，变量声明尽管不是必须的，但在使用变量之前先进行声明是一种好的习惯。可以使用 var 语句来进行变量声明。如：

```
var men = true;   //men 中存储的值为布尔类型
```

在 JavaScript 中，变量声明的一般形式是：

```
var <变量名表>;
```

其中，var 是 JavaScript 中的关键字，表明接下来是变量说明，变量名表是用户自定义的标

识符，变量之间用逗号分开。例如，var a,b,c; 定义了三个变量，没有给出具体的数据类型，也没有赋初值，其数据类型根据初值的数据类型确定。再如，var myName="John";定义了一个变量 myName，同时赋予了它一个字符串值。

（2）变量类型的动态变化　在声明变量时不需要也不能指定变量的类型。JavaScript 解析器会根据变量的当前值以及变量的使用方法确定变量的数据类型，并完成适当的转换。

（3）变量命名　JavaScript 是一种区分大小写的语言，因此将一个变量命名为 computer 和将其命名为 Computer 是不一样的。在对变量命名时，最好把变量的意义与其代表的意思对应起来，以免出现错误。另外，变量名称的长度是任意的，但必须遵循以下规则：

- 第一个字符必须是一个字母或一个下划线（ _ ）或一个美元符号（$）。
- 后续的字符可以是字母、数值、下划线或美元符，不能有空格、+、–或其他符号。
- 不能使用 JavaScript 中的关键字作为变量。在 JavaScript 中定义了 50 多个关键字，这些关键字是 JavaScript 内部使用的，不能作为变量的名称。如 var、int、double、true 不能作为变量的名称。

（4）变量的作用域　变量的作用域由声明变量的位置决定，决定哪些脚本语句可访问该变量。在函数外部声明的变量称为全局变量，其值能被所在 HTML 文件中的任何脚本语句访问和修改。在函数内部声明的变量称为局部变量，局部变量只能被函数内部的语句访问，只对该函数是可见的，而在函数外部则是不可见的。只有当函数被执行时，变量被分配临时空间，函数结束后，变量所占有的空间被释放。

3.2.3　运算符和表达式

1. 运算符

运算符是完成操作的一系列符号，在 JavaScript 中包括以下运算符。

（1）算术运算符　JavaScript 中的算术运算符有单目运算符和双目运算符。

双目运算符：

+（加）、–（减）、*（乘）、/（除）、%（取模）。

单目运算符：

–（取反）、++（自加 1）、– –（自减 1）。

例如：

```
x=10
x++           //x 的值是 11
y=++x-5       //结果是 y 的值为 7,x 的值为 12
y=5+x--       //结果是 y 的值为 17,x 的值为 11
y=x%10        //结果是 x 的值为 11,y 的值为 1
```

（2）关系运算符　关系运算符的基本操作过程是，首先对它的操作数进行比较，然后返回一个 true 或 false 值，下面列出 6 个关系运算符：

<（小于）、>（大于）、<=（小于等于）、>=（大于等于）、==（等于）、!=（不等于）。

例如，设 x=25，y=10，则

```
x<y                    //结果是 false
```

```
x>=y                 //结果是 true
x-15==y              //结果是 true
```

（3）逻辑运算符　包括&&（逻辑与）、||（逻辑或）、!（逻辑非）。基本运算规则见表 3.1。

表 3.1　逻辑运算规则表

A	B	A&&B	A||B	!A
true	true	true	true	false
true	false	false	true	false
false	true	false	true	true
false	false	false	false	true

（4）位操作运算符　包括|（按位或）、&（按位与）、～（按位取反）、^（按位异或）、<<（左移）、>>（右移）、>>>（无符号右移，零填充）。

这些运算符均是面向二进制数位的运算，当进行运算时，首先要转换为二进制数，然后再按二进制数位运算规则进行运算。|、&、^是单个二进制数位的逻辑运算符，与前面介绍逻辑运算符的运算规则类似，此时 1 代表 true，0 代表 false。>>执行算术右移，它使用最高位填充左侧的空位；右移的结果是，每移一位，第一个操作数被 2 除一次，移动的次数由第二个操作数确定。>>>执行逻辑右移，只对位进行操作，而没有算术含义，它用 0 填充左侧的空位。

【例 3.2】 计算下列式子的运算结果。

25&33、18|21、～28、18^21、–35>>2、–35>>>2、32<<2

运算结果分别是：1、23、3、7、–9、55、128。

（5）字符串连接运算符　+。

此运算符的功能是将两个字符串连接起来。下面是该运算的两个式子。

```
"This is a"+"book."      //结果是"This is a book."
"这里共有"+28+"本书。"    //结果是"这里共有 28 本书。"
```

在第二个式子中，运算对象包含了字符串和数值两种类型的数据。由于连接运算会将数值数据类型自动转换为字符串类型，因此可以得到上述结果。

（6）条件运算符　条件运算是一种较为特殊的运算。其语法格式如下：

```
条件? 值 1: 值 2
```

若条件为 true，则结果为值 1，否则结果为值 2。

【例 3.3】 设小于 18 岁为未成年人，大于等于 18 岁为成年人；同时设张三的年龄（age）是 20 岁，即 age=20。请用条件运算判定张三是成年人还是未成年人。

用 result 表示条件运算的结果，年龄小于 18 岁可表示为 age<18，故条件运算可描述为

```
result=age<18?"未成年人":"成年人"     //结果是 result="成年人"
```

2．表达式

在定义完变量后，就可以对它们进行赋值、计算等一系列操作，这些操作通常称为表

达式。可以说它是变量、常量及运算符组合起来的运算式，结果生成一个新的值。表达式可以分为算术表达式、字符串表达式、赋值表达式以及逻辑表达式等。例如，1+2*3 是一个算术表达式；"23"+"56"是一个字符串表达式；a=3 是一个赋值表达式，表示给变量 a 赋值 3。

3.3 JavaScript 程序流程控制语句

正常情况下，程序执行的顺序是按照程序中语句书写顺序执行的。但是，在很多情况下，需要把程序的执行次序转移到另一个位置，这需要条件控制语句实现这一功能。计算机能够快速地重复执行某一些指令，高级语言中使用循环控制语句来描述和实现这一功能。JavaScript 提供了一组条件控制语句和循环控制语句来实现程序流程控制。

3.3.1 条件控制语句

1. if 语句

最简单的条件控制语句是假如某一条件为真，那么程序转向一个特定的程序分支。其基本的语法格式如下：

```
if (条件){
   执行语句
}
```

说明：括号内含有一个运算结果为布尔值的表达式，这就是程序运行到此处要检测的条件。假如条件的值为 true，就执行花括号内的语句组（可以包含任意多条语句），然后继续执行花括号下方的语句；假如条件为 false，则跳过花括号内的语句，程序继续执行括号下方的语句。

下面的例子假设变量 myAge 在脚本中已经预先赋值，条件表达式比较 MyAge 与 18 的大小。

```
if (myAge<18) {
   alert("Sorry, you cannot vote."); //alert()函数的作用是弹出警告对话框
}
```

myAge 的数据类型必须是数值，以便于正确比较得到的结果（通过“<”比较操作符）。对于 myAge 小于 18 的所有实例，执行花括号内的分支语句，并向用户显示警告信息。用户关闭这个警告对话框后，脚本继续处理 if 结构之后的语句。

2. if…else 语句

在 if 结构中，当条件为 false 时不进行任何特殊处理。但是假如处理必须二选一，就需要使用 if…else 语句。其基本语法格式如下：

```
if (条件){
   执行语句 1
} else {
   执行语句 2
}
```

说明：如果其中的条件成立，则程序执行紧接着条件的语句组；否则，程序执行 else 中的语句组。

例如

```
if (result == true){
          response = "你答对了！";
      }else{
          response = "你错了！";
      }
```

3. switch 语句

分支（switch）语句本质上也是一种条件语句，但可以根据一个变量的不同取值采取不同的处理方法。其语法格式如下：

```
switch (表达式){
case 常量 1: 语句组 1
              break;
case 常量 2: 语句组 2
             break;
    ⋮
case 常量 n: 语句组 n
              break;
[default: 语句组 n+1]
}
```

说明：

（1）当变量的值与其中的一个 case 指定的常数相等时，则执行相应的语句组。

（2）当遇到 break 语句时程序跳出 switch 语句，转向执行 switch 语句后面的语句。

（3）如果执行了语句组 k 后，没有遇到 break 语句，那么程序会继续执行语句组 k+1。

（4）当语句中所有的常数都不等于表达式的值时，执行 default 中的语句组。

（5）default 部分可以省略。

【例 3.4】 设某个选举有 4 名候选人，分别用数字 1，2，3，4 表示。假定某人的投票是选择 2。请用 JavaScript 的分支语句判定投票者选择了哪位候选人。

程序描述如下：

```
var vote=2;
switch (vote) {
   case 1:
        document.write("投票人选择了 1 号候选人。");
        break;
   case 2:
        document.write("投票人选择了 2 号候选人。");
        break;
```

```
    case 3:
        document.write("投票人选择了 3 号候选人。");
        break;
    case 4:
        document.write("投票人选择了 4 号候选人。");
}
```

此例中用 vote 表示任意候选人，它的可能情况是 1，2，3，4。vote=1 表示投票选 1，依此类推。

3.3.2 循环控制语句

有时为了使计算机实现某种功能可能要重复执行某一条或一组指令，例如，输出 10 行文字：Welcome to the JavaScript。为此可能需要编写 10 条 document.write 语句。如果要输出 100 或 10 000 行这样的文字又如何呢？显然需要有一种机制来解决这类问题。对于计算机语言来说，解决此类问题的方案就是使用循环控制语句。

循环控制语句提供了几种不同的语法格式。用户可以根据使用场合和习惯来决定使用哪种形式。通常它们有一个共同点，就是通过修改一个变量的值，在进入下一轮循环时判断满足或不满足循环条件来控制循环的次数。

1．for 循环控制语句

语法：

```
for (初始化部分;条件部分;更新部分) {
    执行部分
}
```

说明：

（1）初始化部分　表示循环的初始条件，必须赋予变量初值。

（2）条件部分　是用于判别循环控制变量的值是否满足循环条件。若条件满足，则运行执行部分的语句；否则，跳出循环语句。

（3）更新部分　主要定义循环控制变量在每次循环时按什么方式变化。

（4）三个部分之间，必须使用分号分隔。

【例 3.5】 利用 for 循环控制语句计算 1+2+3+…+100。程序描述如下：

```
var i,result;
result=0;
for(i=1;i<=100;i++){
   result=result+i;
}
```

用 result 表示计算的结果，i 表示加数，且加数有一定的规律，故可采用 for 循环语句实现。

2．while 循环控制语句

语法：

```
while (条件)
    {
```

```
    执行语句
}
```

说明：在该语句中，条件可以是一个逻辑型数据的表达式。整个语句的含义是：当条件的结果为 true 时，运行执行语句；否则，结束循环语句的执行。

【例 3.6】 利用 while 循环语句计算 1＋2＋3＋…＋100。程序描述如下：

```
var i, result;
result=0;
i=1;
while(i<=100){
   result=result+i;
   i++;
}
```

3．do…while 循环控制语句

语法：

```
do {
   执行语句
} while(条件)
```

说明：

（1）由于条件的检测放在语句的最后，因此，执行语句至少被执行一次。

（2）当条件为结果是 true 时，运行执行语句。

【例 3.7】 利用 do…while 语句计算 1×2×3×…×10。程序描述如下：

```
var i,result;
result=1;   // 用 result 表示乘积的结果
i=1;
do{
   result=result*i;
   i++;
} while(i<=10)
```

3.4 JavaScript 函数和事件处理程序

函数为程序设计人员提供了许多方便。通常在设计复杂程序时，总是根据所要完成的功能，将程序划分为一些相对独立的部分，每部分编写为一个函数。从而使程序各部分充分独立，并完成单一的任务，使整个程序结构清晰，达到易读、易懂、易维护的目标。当单击某个按钮、单击某个单选框、将鼠标指针移过某个链接时，都将触发事件。通过编写程序对事件作出反映，这种反映称作响应事件。JavaScript 中，使用函数与特定的事件关联起来，作为事件的处理程序。当发生事件时，与该事件相关联的函数就被执行。

3.4.1 JavaScript 函数

从计算机程序设计的角度来说，函数是一组指令，这组指令被赋予一个名称，这个名称就是函数名。按指定的规则使用函数名就可以执行相应的指令并返回计算结果。在使用

函数时往往需要指定一些相关的数据，这些数据称为函数的参数。使用函数最大的好处就是可以重复利用指令代码，使程序的结构更加清晰。

在程序设计语言中函数通常分为内部函数和自定义函数。内部函数由程序设计语言系统提供，程序员只要了解其名称、参数和返回的结果等便可以使用这些函数来完成指定的任务。自定义函数则需要程序员自己在使用前对函数的名称、参数、具体执行的指令和返回结果进行说明，然后才可以使用该函数。对自定义函数的说明在术语上称为函数定义，使用函数则称为调用函数。

如果想教会自己快速阅读并且用一个一旦单击可告诉你当前时间的长文本链接。可以用程序实现如下：

```
<a href="#" onClick=" ">time!</a>
var the_date = new Date();
var the_hour = the_date.getHours();
var the_minute = the_date.getMinutes();
var the_second = the_date.getSeconds();
var the_time = the_hour + ':' + the_minute + ':' + the_second;
alert('The time is now: ' + the_time);
```

在这里这段 JavaScript 的工作细节并不重要，重要的是它太长了。若这些时间链接再有 10 个，就须每次复制这段程序。这使 HTML 既长且难看。另外，若想改变这段程序，就必须在 10 个不同地方改变。好的办法是：定义一个函数来执行而不用作 10 次复制程序。

函数定义语法如下：

```
function 函数名（参数表）{
    语句组
    return  结果
}
```

说明：

（1）函数名的命名规则与变量名的命名规则相同。

（2）函数可以没有参数，但仍然需要在函数名之后指定圆括号。

（3）函数不一定需要返回计算结果。如果需要返回结果，则需要使用 return 语句并在其后指定返回的结果；如果不用返回结果无须使用 return 语句。

（4）在函数中，当执行到 return 语句或遇到右大括号（}）时函数便结束。

（5）尽管在 HTML 文件中的大部分地方都可以定义函数，但是在<head></head>标记之间定义函数是一个比较好的做法。

（6）调用函数时只要指定函数的名称，并将具体的参数按定义时的顺序填写在圆括号中即可。如果函数调用时没有参数则仍然需要输入一对空的圆括号。

【例 3.8】 编写一个网页文件 ex3_8.htm，使用 JavaScript 定义一个函数用于计算两个整数的和。

网页文件的内容如下：

```
<HTML>
```

```
<HEAD>
<TITLE>计算两个整数的和 </TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!--
function sumXY(x,y){    //定义函数
    var result;
    result=x+y;
    return result;
  }
 //-->
</SCRIPT>
</HEAD>
<BODY>
<font size=5>
<SCRIPT LANGUAGE="JavaScript">
<!--
var r;
r=sumXY(250,150);
document.write("250+150=",r);
//-->
</SCRIPT>
</BODY>
```

</HTML>上述代码中，在<head></head>标记之间定义求和函数 sumXY(x,y)，用来求参数 x 与 y 的和，并返回结果。通过语句 r=sumXY(250,150)调用函数 sumXY(x,y)，并把返回的值赋给变量 r。最后通过语句 document.write("250+150=",r)把结果输出。

在浏览器中打开文件，结果如图 3.3 所示。



图 3.3　函数定义的示意图

3.4.2　JavaScript 事件处理程序

事件是浏览器响应用户交互操作的一种机制，JavaScript 的事件处理机制可以改变浏览器响应用户操作的方式，这样就开发出具有交互性，并易于使用的网页。

事件定义了用户与页面交互时产生的各种操作，例如，单击超级链接或按钮时，就会产生一个单击（click）操作事件。浏览器为了响应某个事件而进行的处理过程，叫做事件处理。浏览器在程序运行的大部分时间都等待交互事件的发生，并在事件发生时，自动调用事件处理程序，完成事件处理过程。设置一个对象使其在某个事件发生时便执行指定的函数，需要遵循指定的语法格式。格式如下：

```
on 事件名="事件处理代码"
```

事件不仅可以在用户交互过程中产生，而且浏览器自己的一些动作也可以产生事件，例如，当载入一个页面时，就会发生 load 事件，卸载一个页面时，就会发生 unload 事件等。

JavaScript 中常见的事件见表 3.2。

表 3.2　JavaScript 中常见的事件

事件名	触发事件的条件	事件名	触发事件的条件
Blur	对象失去焦点	Mousedown	在对象处按下鼠标按键
Change	对象中的内容发生变化	Mousemove	鼠标在对象表面移动
Click	单击对象	Mouseover	鼠标在对象表面
Dblclick	双击对象	Mouseout	鼠标移离对象表面
Focus	对象获得焦点	Mouseup	鼠标键被释放
Keypress	发生键盘击键	Move	窗口被移动
Keydown	键盘的按键被按下	Resize	窗口被放大或缩小
Keyup	键盘的按键被释放	Submit	表单内容被执行提交
Load	浏览器窗口中载入文档内容	Select	文本框或文本区域中的文本被选定
Unload	关闭浏览器中的文档		

在表 3.2 中提到的对象获得焦点的含义是指该对象可以接收键盘输入。例如，对于文本框来说，只有当其获得焦点时用户才可以在其中输入内容。

【例 3.9】 当浏览器中文档的主题部分<body>被单击时，会产生 click 事件。请编写一个网页文件 ex3_9.htm，使得用户单击浏览器窗口中的文档时，显示一个消息框。

网页文件的内容如下：

```
<HTML>
<HEAD>
<TITLE>事件响应实例</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!--
   function sayThanks(){
      alert("谢谢你的单击");
   }
//-->
</SCRIPT>
</HEAD>
<BODY onClick="sayThanks()">
<H1  align=center>请单击一下</H1>
</BODY>
</HTML>
```

在浏览器中打开文件，结果如图 3.4 所示。

在该网页文件中的 body 标记中设置了 onClick="sayThanks()"。这样，当文档主体被单击时，sayThanks()函数便会自动执行，从而利用 alert()函数显示出消息框。当单击浏览器窗口的主体部分时，显示指定内容的消息框，如图 3.5 所示。

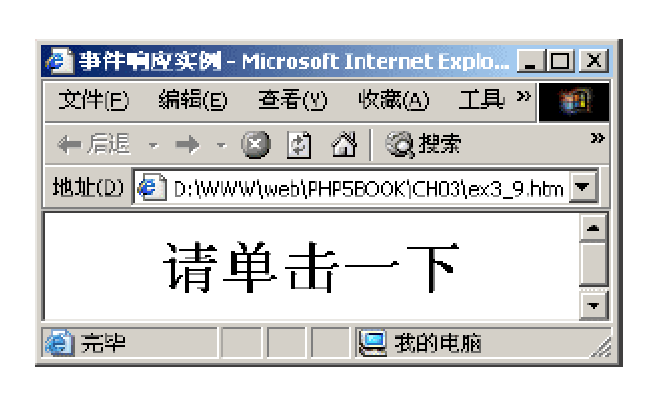


图 3.4　打开网页文件后的窗口



图 3.5　单击窗口后的消息框

3.5　HTML 文档对象模型

JavaScript 是一种基于对象的编程语言。除了语言本身提供了一组内部对象可供使用外（将在 3.6 节中介绍），浏览器也提供了一组可以操作的对象，可以制作出更精彩的网页。

3.5.1　对象的概念

什么是对象？从日常生活中来看，汽车是一个对象，书是一个对象，人是一个对象。对于计算机系统来说，几乎屏幕上见到的所有东西都可以看成对象。例如，一个应用程序窗口、一个按钮、一个菜单，等等。

为了描述一个对象，通常需要描述该对象的若干静态特征和该对象的若干动态行为。例如，描述汽车这个对象时，它的特征包括汽车的颜色、大小、座位数等；汽车的行为包括启动、前进、后退、拐弯、停止等。

在面向对象概念中，通常把对象的静态特征叫做对象的属性，把对象的动态行为叫做对象方法。

综上所述，对象被看成具有某些属性和方法的逻辑实体。JavaScript 中的对象也是由属性和方法两个基本元素构成的。对象的属性是指对象的本质特征；对象的方法是指对属性所进行的操作，就是一个对象自己所有的函数。

可以采用下面的方式来访问对象的属性、方法：

```
对象名称.属性名称
对象名称.方法名称
```

例如，

```
mycomputer.year=1996， mycomputer.owner = "me"
document.writeln ("this is method")
```

JavaScript 语言是基于对象的，而不是面向对象的。之所以说它是一门基于对象的语言，主要是因为它没有提供像抽象、继承、重载等有关面向对象语言的许多功能。而是把其他语言所创建的复杂对象统一起来，从而形成一个非常强大的对象系统。

虽然在 JavaScript 中对象的功能已经是非常强大了，但更有用的是设计人员可以按照需求来创建自己的对象，以满足某一特定的要求。使用 New 运算符可以创建一个新的对象。其创建对象使用如下格式：

```
Newobject=new Object(参数表)
```

其中，Newobject 是创建的新对象，Object 是已经存在的对象；参数表含有 0 个以上的参数；new 是 JavaScript 中的关键词，用于创建一个新的对象。例如，创建一个日期新对象：

```
newDate=new Date();
```

或

```
birthday=new Date("December 12,2006");
```

之后就可使 newDate 或 birthday 作为一个新的日期对象了。

3.5.2 HTML 文档对象模型

由于 JavaScript 是在浏览器中执行的一种语言，因此，它还可以利用浏览器内部提供的对象对浏览器进行各种操作。浏览器的结构是基于对象的结构，整个结构由各种对象组成，呈现树状。通常这样一个模型称为文档对象模型（DOM），文档对象模型就是所有那些对象组成的总体，是一种为了命名所有对象的系统，该命名系统是建立在对象层次的基础上。任意对象都在 DOM 中，其主要结构如图 3.6 所示。

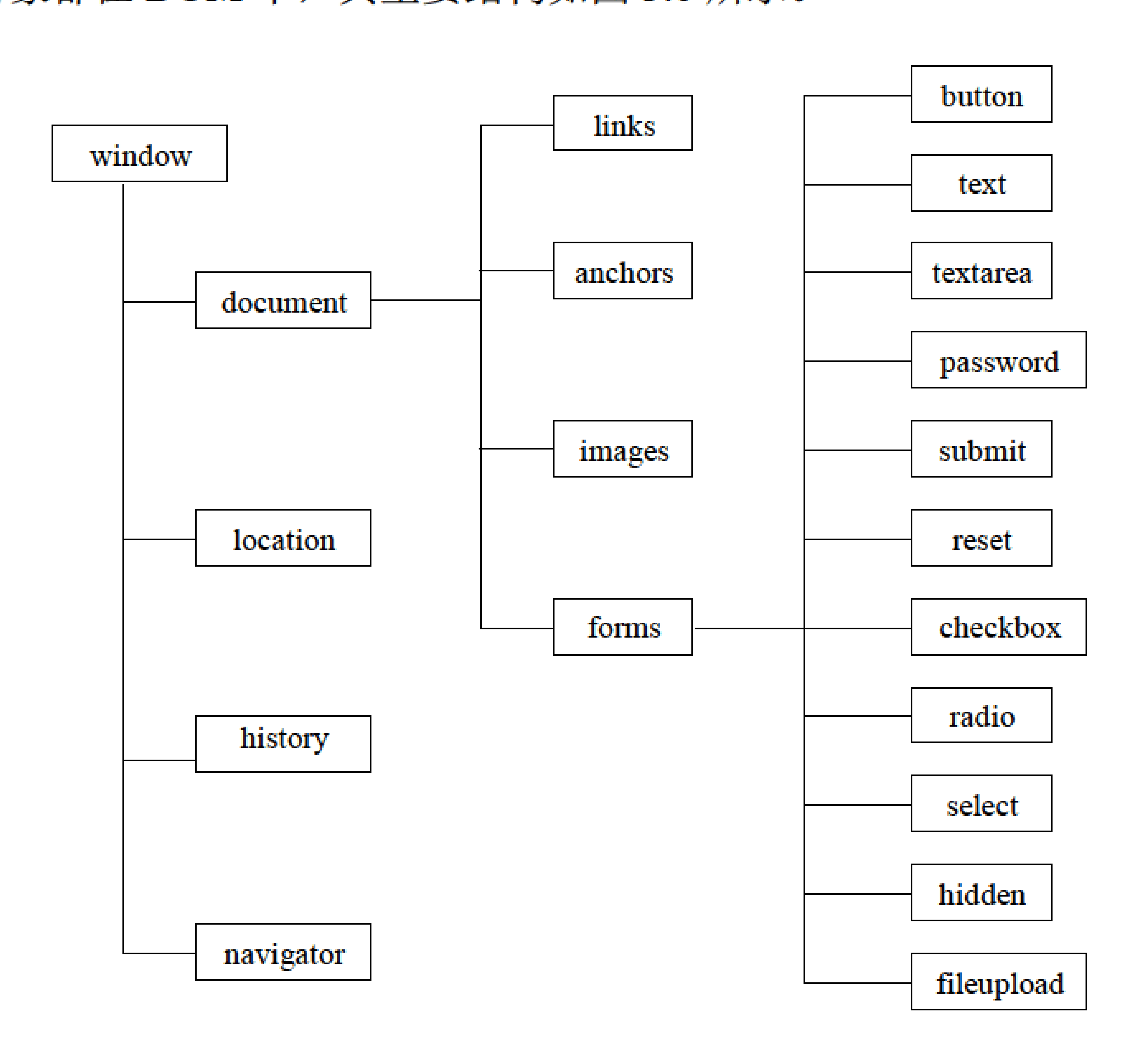


图 3.6 浏览器对象的主要结构

1．window 对象

window 对象是最顶级的对象，它是其他对象的父对象。这种对象代表 HTML 文档在浏览器窗口的内容区域。在多重框架环境下，每个框架都是一个窗口。所有的动作都是在窗口内发生的，窗口是对象层次中最外部的元素，它的物理界限包含文档。

window 对象提供了许多属性和方法，利用这些属性、方法，再配合相应的时间处理就可以实现浏览器窗口的许多功能，开发出既美观，交互性又好的页面。

（1）window 对象的常用属性及功能如下：

- document　表示窗口中显示的当前文档。
- history　表示最近访问过的 URL 列表。
- location　表示窗口中显示的当前 URL。
- status　表示窗口状态栏中显示的信息。
- defaultStatus　表示窗口状态栏中显示的默认信息。

（2）window 对象的常用方法及功能如下：

- alert()　它是一个弹出对话框，用以提示用户某些注意事项。
- confirm()　它是一个消息框，有“确认”和“取消”两个按钮，单击“确认”按钮返回 true，单击“取消”按钮，返回 false，利用返回值可以作进一步的工作。
- prompt()　这是一个消息框，不过该消息框允许用户输入某种信息。该信息被当作 prompt()的结果返回。
- open()　该方法打开一个新的浏览器窗口，原窗口不变。新打开的窗口可以定义大小、有无工具栏、有无状态栏、有无地址栏、可否改变尺寸、有无滚动条。
- close()　关闭当前浏览的窗口。
- blur()　从窗口中移走焦点，在很多系统中，该操作把窗口送往后台。
- focus()　使窗口获得焦点，在很多系统中，该操作把窗口送往前台。

【例 3.10】 编写一个网页文件 ex3_10.htm，当打开该网页时，在浏览器窗口的状态栏中从右到左滚动显示文字“欢迎光临本购物网站”。

网页文件的内容如下：

```
<HTML>
<HEAD>
<TITLE>WINDOW 范例</TITLE>
</HEAD>
<BODY>
<center><h1>这里是购物网站,欢迎您的光临!</h1></center>
<SCRIPT LANGUAGE="JavaScript">
<!--
     var msg="  ...欢迎光临本购物网站!...    ";
     var pos=0;
     function scrollMsg(){
window.status=msg.substr(pos,msg.length)+msg.substr(0,pos);
       pos++;
       if(pos>msg.length) pos=0;
     }
     setInterval("scrollMsg()",300);   //定时执行函数
//-->
</SCRIPT>
```

```
</BODY>
</HTML>
```

在浏览器中打开文件，结果如图 3.7 所示。

在该网页文件中的 body 标记中用定时执行函数 setInterval()定时调用 scrollMsg()函数。scrollMsg()函数每 300 毫秒执行一次，每次在状态栏的某个指定位置显示“…欢迎光临本购物网站!…”，从而实现在浏览器窗口的状态栏中从右到左滚动显示文字，如图 3.7 所示。

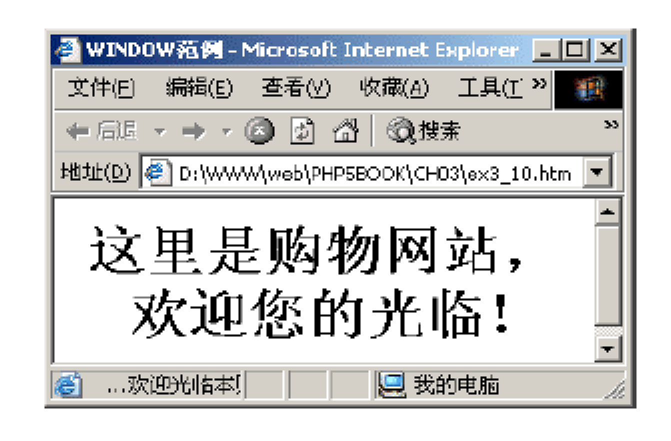


图 3.7　状态栏中从右到左滚动显示文字

2．document 对象

窗口中的内容是 document，因此它是最主要的对象之一。每个载入窗口中的 HTML 文档成为一个 document 对象。document 对象几乎包含了模型中最多的其他类对象。可以这样认为：文档包括了用户想要编辑的全部内容。该对象提供对浏览器窗口中文档属性的管理和文档内容的读写等功能。文档对象的基本属性和方法如下。

（1）document 对象的常用属性及功能如下：

- bgColor　页面的背景色。
- fgColor　页面的前景色，即文本的颜色。
- linkColor　超文本链接的颜色。
- alinkColor　鼠标指向的超链接文本的颜色。
- vlinkColor　已经单击过的超链接文本的颜色。
- lastModified　此页面（HTML 文件）最后被修改的时间。
- forms　表单（form）对象组成的数组，数组中的每一个元素对应于网页中的每一个<FORM>标记，数组元素对应的顺序是在 HTML 文件代码中<FORM>标记出现的先后顺序。
- links　超文本链接对象组成的数组，数组中的每一个元素对应于网页中的每一个<A>标记，数组元素对应的顺序是在 HTML 文件代码中<A>标记出现的先后顺序。
- title　当前页面的标题。

从以上属性描述可以看出，document 对象的相当一部分属性与 HTML 中 BODY 标记的属性类同。因此，也可以通过设置 BODY 标记的属性来达到同样的目的。

（2）document 对象的常用方法。

document 对象最常用的方法为 write()，表示在文档中写内容。使用时的格式为：

```
document.write(参数 1,参数 2,…,参数 n)
```

其中，所有的参数都会作为 HTML 文件的内容传输到浏览器中；而且多个参数也可以通过“+”运算连接成为一个参数。

【例 3.11】 编写一个网页文件 ex3_11.htm。要求单击页面时可以显示一个输入提示框，让用户输入一种颜色值；然后，将其设置为网页的背景色。

网页文件的内容如下：

```
<HTML>
<HEAD>
```

```
<TITLE>DOCUMENT 范例</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!--
  function setBkColor(){
     var newColor=prompt("请输入新的颜色值","3344ee");
     if(newColor!=null){
         document.bgColor=newColor;
     }
  }
//-->
</SCRIPT>
</HEAD>
<BODY onClick="setBkColor()">
<center><h1>这里是购物网站,欢迎您的光临!</h1></center>
</BODY>
</HTML>
```

在浏览器中打开网页文件后，如图 3.8 所示。单击页面后，出现如图 3.9 所示的窗口，然后输入一个颜色值，或使用默认的颜色值，最后确认。背景被设置为指定的颜色。值得一提的是颜色值可以为十六进制数，也可以是命名的颜色（如 red）。

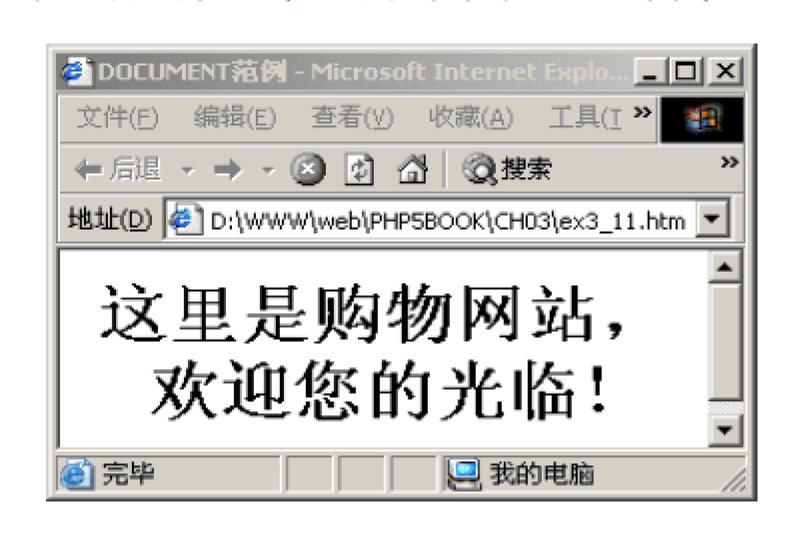


图 3.8　打开网页



图 3.9　弹出的输入提示框

3．form 对象

form 对象封装了<form>标签定义的表单中的相关信息，主要用于收集用户的数据信息。用户在网页中看不见表单在何处开始和结束，只能看到它们的元素。但是表单在一个 HTML 文档中是很特殊的一组内容，在<FORM>…</FORM>标记对中所有的内容都是 form 对象的一部分。如果是设计需要，一个文档可以包含不止一对<FORM>标记。

它的几个数值属性基本是和 HTML 的<form>标签中的属性相对应，有 type、name、action、method、encoding、target 等。除了上面和 HTML 标签相对应的属性外，还有两个属性是和表单内的元素有关的，一个是 elements，它是一个数组，数组中的每一个元素都是由<input>定义的一个对象；另一个是 length，它是 elements 数组的元素的个数。

form 对象的方法较少，只有 submit()和 reset()两个。submit()的功能是将表单提交给服务器中指定的处理程序，reset()功能是清空表单中的内容让用户重新输入信息。

在学习第 2 章的 HTML 知识时已经知道如何在网页中建立一个表单。在 3.7 节中，将介绍如何在表单提交到服务器之前获取表单数据并对其进行有效性检查和必要的计算

处理。

4．Location 对象

location 对象是 window 对象的一个属性。在浏览器对象结构中处于 window 对象的下一层。该对象的主要功能是提供与 window 对象相关的 URL 的完整信息。location 对象是一个静态的对象，它提供了与当前打开的 URL 一起工作的属性和方法。

（1）location 对象的属性见表 3.3。

表 3.3　location 对象的属性

属性	功能说明	属性	功能说明
hash	指定 URL 中一个锚点名称	pathname	URL 的路径部分
host	指定主机的域名或 IP 地址	port	服务器所用的通信端口
hostname	主机名称和端口号，中间用冒号分隔	protocol	URL 的协议部分（含冒号）
href	完整的 URL	search	提交给服务器的搜索信息

（2）location 对象的方法见表 3.4。

表 3.4　location 对象的方法

方法	说明
reload()	强制重新载入当前窗口中的文档
replace(URL)	用指定的 URL 的文档替换当前窗口中文档

【例 3.12】 编写一个网页文件 ex3_12.htm，在浏览器打开该网页文件时，自动转向显示另外一个指定的网页。

网页文件的内容如下：

```
<HTML>
<HEAD>
<TITLE>location 对象范例</TITLE>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
<!--
  location.href="http://www.sina.com.cn";
//-->
</SCRIPT>
</BODY>
</HTML>
```

在浏览器中打开网页文件后，浏览器中的内容被自动转向载入新浪网站的主页。这种技巧常常用于这样的场合：一个已被广大用户熟知的网页文件位置发生了变化，应用这种方法使用户在无通知的情况下，仍然能够访问到新位置的网页。

3.5.3　引用 HTML 对象

网页文件是由 HTML 标记组成的，这些标记中的大部分都可以作为对象来处理。为了

对 HTML 标记对象进行操作，需要有一种引用标记对象的方法。为此，要为那些需要进行动态处理的标记设置它的 ID 属性。一旦设置了该属性，就可以在 JavaScript 中通过这个 ID 的属性值来对该标记对象进行操作。

由于网页中的标记可以作为对象在程序中进行处理，这就为制作一个生动活泼、多姿多彩、功能强大的网页提供了无限的想象空间。

【例 3.13】 编写一个网页文件 ex3_13.htm，使其中的大标题会自动地以红、绿、蓝、黄的顺序反复改变颜色。

网页文件的内容如下：

```
<HTML>
<HEAD>
<TITLE>HTML 标记对象范例</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!--
  var n=0;
  function changeFontColor(){
     n=n%4;
     switch(n){
       case 0:
         main_title.color="red";
         break;
       case 1:
         main_title.color="green";
         break;
       case 2:
         main_title.color="blue";
         break;
       case 3:
         main_title.color="yellow";
     }
     n++;
  }
  //定时执行函数每秒钟调用 changeFontColor()函数一次,改变大标题的颜色
setInterval("changeFontColor()",1000);
//-->
</SCRIPT>
</HEAD>
<BODY>
<font ID=main_title size=6><b>虚拟网络世界</b></font>
</BODY>
</HTML>
```

由于给字体标记 font 的 ID 属性进行了赋值，因此，在 JavaScript 的代码中就可以利用

该 ID 对字体对象进行操作，从而实现对字体颜色的修改。在浏览器中打开文件，结果如图 3.10 所示。

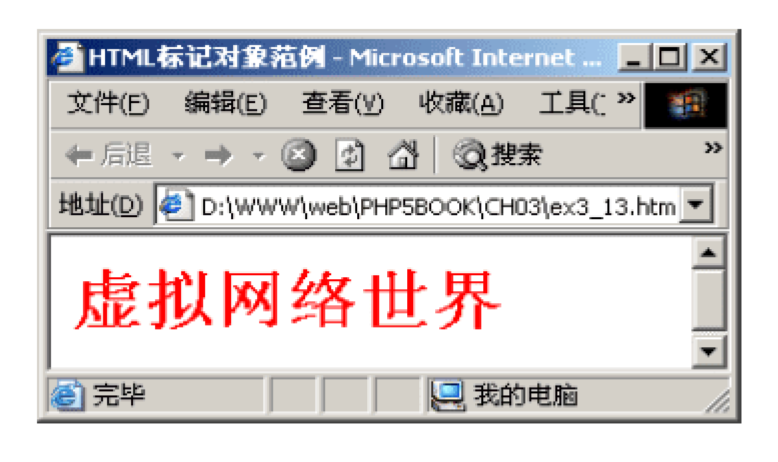


图 3.10 不断变色的网页

3.6 JavaScript 内置的常用对象

JavaScript 的内置对象包括 String、Math 和 Date 等。这些对象为程序设计提供了现成的工具，从而大大降低了编码的复杂性。JavaScript 的内置对象主要是为程序处理提供便利，这些对象基本上不提供对事件的响应。

3.6.1 String 对象

String 对象是在程序编码中应用最为广泛的一个对象。它主要提供字符串的各种属性和处理字符串的各种方法，从而简化在编码过程中对字符串所做的各种处理。

声明一个字符串对象最简单、快捷、有效、常用的方法就是直接赋值。例如：

```
var str1="我是一个学生";
```

另外，还可以采用以下方法声明一个字符串对象。如：

```
var str2=new  String("我是一个学生");
```

1．String 对象的属性

String 对象的主要属性是 length，该属性的功能是返回该字符串中的字符个数，包括所有符号。例如，字符串“This is a JavaScript”的长度是 20。同样，对于中文则是一个汉字作为一个字符计算。例如，字符串“我喜欢网页设计”的长度是 7。

2．String 对象的方法

String 对象提供了大量的方法以方便对字符串的各种处理。主要用于有关字符串在 Web 页面中的显示、字体大小、字体颜色、字符的搜索以及字符的大小写转换。其主要方法如表 3.5 所示。

【例 3.14】 编写一个网页文件 ex3_14.htm，用斜体字输出字符串：勾股定理。然后在下一行中输出公式 $a^2+b^2=c^2$。所有字体大小采用 5 号字。

网页文件的内容如下：

```
<HTML>
<HEAD>
```

```
<TITLE>勾股定理</TITLE>
</HEAD>
<BODY>
<font size=5>
<SCRIPT LANGUAGE="JavaScript">
<!--
      var mstr=new String("勾股定理");
      document.write(mstr.italics(),"<br>");
document.write("a"+"2".sup()+"+b"+"2".sup()+"=c"+"2".sup());
//-->
</SCRIPT>
</font>
</BODY>
</HTML>
```

表 3.5　String 对象的主要方法

方法	用法	功能
charAt()	<字符串对象>. charAt(<位置>)	返回该字符串位于第<位置>位的单个字符。注意：字符串中的第一个字符是第 0 位的，第二个才是第 1 位的，最后一个字符是第 length –1 位的
charCodeAt()	<字符串对象>. charCodeAt(<位置>)	返回该字符串位于第<位置>位的单个字符的 ASCII 码
indexOf()	<字符串对象>. indexOf(<另一个字符串对象>[, <起始位置>])	该方法从<字符串对象>中查找<另一个字符串对象>（如果给出<起始位置>就忽略之前的位置），如果找到了，就返回它的位置，没有找到就返回“–1”。所有的“位置”都是从零开始的
substring()	<字符串对象>. substring(<始>[, <终>])	返回原字符串的子字符串，该字符串是原字符串从<始>位置到<终>位置的一段。<终>–<始>=返回字符串的长度（length）。如果没有指定<终>或指定的超过字符串长度，则子字符串从<始>位置一直取到原字符串尾。如果所指定的位置不能返回字符串，则返回空字符串
substr()	<字符串对象>. substr(<始>[, <长>])	返回原字符串的子字符串，该字符串是原字符串从<始>位置开始，长度为<长>的一段。如果没有指定<长>或指定的超过字符串长度，则子字符串从<始>位置一直取到原字符串尾。如果所指定的位置不能返回字符串，则返回空字符串
toLowerCase()	<字符串对象>. toLowerCase()	返回把原字符串所有大写字母都变成小写的字符串
toUpperCase()	<字符串对象>. toUpperCase()	返回把原字符串所有小写字母都变成大写的字符串
bold()	<字符串对象>. bold()	用于把字符串以粗体字形式输出
italics()	<字符串对象>. italics()	用于把字符串以斜体字形式输出
sub()	<字符串对象>.sub()	用于把字符串以下标字符形式输出
sup()	<字符串对象>.sup()	用于把字符串以上标字符形式输出

在浏览器中打开文件，结果如图 3.11 所示。



图 3.11 设置字符串的格式

3.6.2 Math 对象

Math 对象，提供除加、减、乘、除以外的一些运算。如对数、平方根等。但是，Math 对象与前面介绍的对象有一个明显的不同之处，就是 Math 对象不需要像其他对象那样使用 new 关键字生成对象实例。在使用 Math 对象时，只要直接使用 Math 作为对象名即可。

1. Math 对象的属性

下面是 Math 对象的属性和它们的说明。

- E 返回常数 e（2.718281828…）。
- LN2 返回 2 的自然对数（ln 2）。
- LN10 返回 10 的自然对数（ln 10）。
- LOG2E 返回以 2 为底的 e 的对数（lb e）。
- LOG10E 返回以 10 为底的 e 的对数（lg e）。
- PI 返回 π（3.1415926535…）。
- SQRT1_2 返回 1/2 的平方根。
- SQRT2 返回 2 的平方根。

2．Math 对象的方法

下面是 Math 对象的常用方法及说明。

- abs(x) 返回 x 的绝对值。
- acos(x) 返回 x 的反余弦值（余弦值等于 x 的角度），用弧度表示。
- asin(x) 返回 x 的反正弦值。
- atan(x) 返回 x 的反正切值。
- atan2(x, y) 返回复平面内点（x, y）对应的复数的幅角，用弧度表示，其值在–π 到 π 之间。
- ceil(x) 返回大于等于 x 的最小整数。
- cos(x) 返回 x 的余弦。
- exp(x) 返回 e 的 x 次幂（e^x）。
- floor(x) 返回小于等于 x 的最大整数。
- log(x) 返回 x 的自然对数（ln x）。
- max(a, b) 返回 a，b 中较大的数。
- min(a, b) 返回 a，b 中较小的数。
- pow(n, m) 返回 n 的 m 次幂（n^m）。
- random() 返回大于 0 小于 1 的一个随机数。
- round(x) 返回 x 四舍五入后的值。
- sin(x) 返回 x 的正弦。
- sqrt(x) 返回 x 的平方根。

- tan(x)　返回 x 的正切。

【例 3.15】 编写一个网页文件 ex3_15.htm，实现一个抽奖程序：页面中连续快速随机显示 1000～3000 之间的 4 位数号码，当按下任意一个键时停止并显示抽中的数值。

网页文件的内容如下：

```
<HTML>
<HEAD>
<TITLE>抽奖程序</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!--
   var r;
  function showNextNum(){
     var m_num=Math.floor(Math.random()*(3000-1000))+1000;
     num.innerHTML=m_num;
  }
  function showCapture(){
     clearTimeout(r);
  }
//-->
</SCRIPT>
</HEAD>
<BODY onKeypress="showCapture()">
<H1 ID=num  align=center>0000</H1>
<SCRIPT LANGUAGE="JavaScript">
<!--
    r=setInterval("showNextNum()",100);
//-->
</SCRIPT>
</BODY>
</HTML>
```

在浏览器中打开该文件，按下任意一个键，结果如图 3.12 所示。这里使用定时执行函数 setInterval()系统定时执行 showNextNum 函数。每次执行该函数时随机产生一个 1000～3000 之间的整数，并通过 ID 为 num 的 H1 标记显示出来。

为了能够在按下任意键时停止，程序中为 BODY 标记设置了 Keypress 事件的处理代码。利用 clearTimeout()函数停止定时执行任务，从而实现抽奖目的。



图 3.12　按下任意一个键后选中的号码

3.6.3　Date 对象

在编写网页的过程中，页面内容往往涉及日期和时间。日期对象 Date 可以提供日期、时间方面的详细信息。尽管该对象称为日期对象，但是它包含时间信息。

声明一个日期对象有两种方法。

1．var 变量名= new Date()

例如，var d = new Date()，这个方法使 d 成为日期对象，并且已有初始值：当前时间。

2．var 变量名= new Date **（日期参数）**

这种方法可以用日期参数指定的日期、时间来产生日期对象。例如：

```
var d = new Date(06, 10, 1)  //2006 年 10 月 1 日
var now = new Date("Jan 20,2006 9:30:00")
```

由于日期对象只是用于提供和设置日期和时间方面的信息，因此它只有方法而没有属性。表 3.6 是 Date 对象常用的获得/设置日期和时间的方法。以下有很多“g/set[UTC]XXX”这样的方法，它表示既有“getXXX”方法，又有“setXXX”方法。“get”是获得某个数值，而“set”是设定某个数值。

表 3.6　Date 对象常用的获得/设置日期和时间的方法

g/setFullYear()	返回/设置年份，用四位数表示，如果使用“x.setFullYear(99)”，则年份被设定为 0099 年
g/setYear()	返回/设置年份，用两位数表示，设定的时候浏览器自动加上“19”开头，故使用“x.setYear(00)”把年份设定为 1900 年
g/setMonth()	返回/设置月份
g/setDate()	返回/设置日期
g/setDay()	返回/设置星期，0 表示星期天
g/setHours()	返回/设置小时数，24 小时制
g/setMinutes()	返回/设置分钟数
g/setSeconds()	返回/设置秒钟数
g/setMilliseconds()	返回/设置毫秒数
g/setTime()	返回/设置时间，该时间就是日期对象的内部处理方法：从 1970 年 1 月 1 日零时正开始计算到日期对象所指的日期的毫秒数。如果要使某日期对象所指的时间推迟 1 小时，就用“x.setTime(x.getTime() + 60 * 60 * 1000);”（一小时 60 分，一分 60 秒，一秒 1000 毫秒）
getTimezoneOffset()	返回日期对象采用的时区与格林尼治时间所差的分钟数，在格林尼治东方的市区，该值为负，例如，中国时区（GMT+0800）返回“–480”
toString()	返回一个字符串，描述日期对象所指的日期，这个字符串的格式类似于“Fri Jul 21 15:43:46 UTC+0800 2000”
toLocaleString()	返回一个字符串，描述日期对象所指的日期，用本地时间表示格式。如“2000-07-21 15:43:46”
parse()	返回该日期对象的内部表达方式，用法：Date.parse(<日期对象>)

【例 3.16】 编写一个网页文件 ex3_16.htm，在网页中显示一个数字式电子时钟。

网页文件的内容如下：

```
<HTML>
```

```
<HEAD>
<TITLE>数字式电子时钟 </TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!--
    function setTime(){
      var now=new Date()
      timer.innerHTML=now.getHours()+":"+now.getMinutes()+":"
      +now.getSeconds()
    }
//-->
</SCRIPT>
</HEAD>
<BODY>
<H1 ID=timer  align=center>00:00:00</H1>
<SCRIPT LANGUAGE="JavaScript">
<!--
    setInterval("setTime()",1000)
//-->
</SCRIPT>
</BODY>
</HTML>
```

在浏览器中打开该文件，结果如图 3.13 所示。这里利用了标记对象，在代码中，H1 标记的 ID 属性被赋予属性值：timer。这样，就可以在代码中修改 H1 标记中显示的内容。此外，利用 setInterval()函数设置每秒钟执行 1 次 setTime()函数以更新时间，更新的时间通过每次产生的 Date 对象获取。

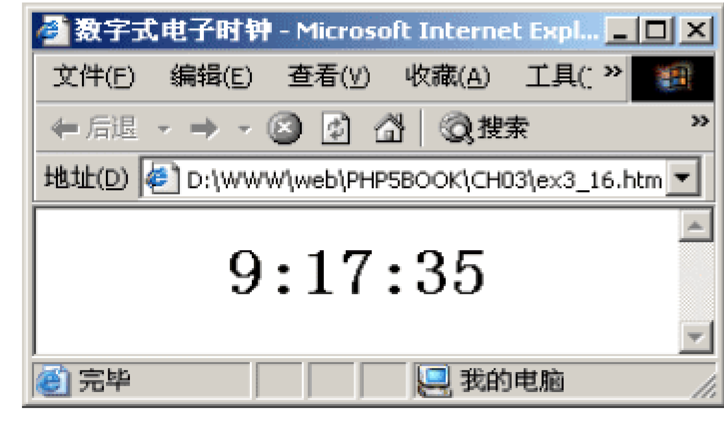


图 3.13　在网页中显示系统时间

3.7　用 JavaScript 脚本验证 HTML 数据

用 JavaScript 脚本可以对 HTML 中的数据进行确认或预先验证。如果验证发现了一些不正确的数据或空白域，那么就可以根据验证函数的结果取消提交。

3.7.1　显示消息

在 JavaScript 的内部函数中，有几个函数是用于产生消息框和接收输入内容的输入框。消息框通常用于提醒用户执行某种操作，并提供若干选择让用户作出响应。输入提示框用于接收用户的一些简单的键盘输入字符。

1．alert()函数

alert()函数是 window 对象的一个方法，因此在使用时，不需要写 window 窗口对象名，而是直接使用就行了。其主要用途是显示一个带有提示或警告字符串内容信息框，一旦用户单击“确定”按钮，方可继续执行其他脚本程序。

【例 3.17】 编写一个网页文件 ex3_17.htm，当用户使用浏览器打开该程序时自动显示一个消息框并在上面显示：这是一个 JavaScript 测试程序。

网页文件的内容如下：

```
<HTML>
<HEAD>
<TITLE>alert 函数范例</TITLE>
</HEAD>
<BODY>
<Script Language ="JavaScript">
alert("这是一个 JavaScript 测试程序");
</Script>
</BODY>
</HTML>
```

在浏览器中打开该文件，结果如图 3.14 所示。

2．confirm()函数

confirm()函数与 alert()类似，但是它提供“确定”和“取消”两种选择让用户作出响应。它的主要用途是显示一个带有“确定”和“取消”两种按钮的消息框，当用户单击“确定”按钮时，该函数返回逻辑值 true；否则，返回 false。只有用户作出选择后，即单击“确定”或“取消”按钮，才能继续执行其他脚本程序。

【例 3.18】 编写一个网页文件 ex3_18.htm，当用户打开该网页时，允许选择蓝色或黑色的标题。

网页文件的内容如下：

```
<HTML>
<HEAD>
<TITLE>confirm 函数范例</TITLE>
</HEAD>
<BODY>
<Script Language ="JavaScript">
   var echo=confirm("要显示蓝色标题吗? ");
   if(echo==true)
      document.write("<font color=blue><h1>这是一个测试程序</h1> </font>");
   else
     document.write("<h1>这是一个测试程序</h1>");
</Script>
</BODY>
</HTML>
```

在浏览器中打开文件，结果如图 3.15 所示。



图 3.14　使用 alert 函数确认信息



图 3.15　使用 confirm 函数验证

3.7.2 打开新的浏览器窗口

打开一个新的浏览器窗口需要使用 Window 对象的open()方法。open()方法的语法格式是：

```
Window.open(url,窗口名称,窗口属性)
```

说明：

（1）url 参数是一个字符串，用于指定在新窗口中打开的文件。如果指定一个空串，则只是打开一个空白窗口。例如，要打开一个新窗口用于显示新浪网站的主页，则可以使用下面的语句。

```
Window.open("www.sina.com.cn");
```

（2）窗口属性参数用于指定新窗口的特征。这些属性是可以同时设置的，同时设置多个属性时用逗号分隔每个属性。表 3.7 是该方法可用的属性和可能的取值。

表 3.7　窗口属性参数可用的属性和可能的取值

属性	取值	说明
Memubar	yes/no	是否出现菜单栏
Scrollbar	yes/no	是否出现滚动条
Status	yes/no	是否出现状态栏
Toolbar	yes/no	是否出现工具栏
Resizable	yes/no	是否可以改变窗口大小
Width	像素值	设置窗口的宽度
Height	像素值	设置窗口的高度

【例 3.19】 编写一个网页文件 ex3_19.htm。当打开该网页时，在浏览器窗口的状态栏中从右到左滚动显示文字：“欢迎光临本购物网站”；同时打开一个宽度为 400，高度为 300，不含菜单栏和工具栏的新窗口，并在其中显示文字：优惠促销正在进行中。

网页文件的内容如下：

```
<HTML>
<HEAD>
<TITLE>WINDOW 范例</TITLE>
</HEAD>
<BODY>
<center><h1>这里是购物网站,欢迎您的光临!</h1></center>
<SCRIPT LANGUAGE="JavaScript">
<!--
  var msg="  ...欢迎光临本购物网站!...    ";
  var pos=0;
  //滚动字符
function scrollMsg(){
    window.status=msg.substr(pos,msg.length)+msg.substr(0,pos);
```

```
        pos++;
        if(pos>msg.length) pos=0;
     }
     setInterval("scrollMsg()",300);
     //打开一个新的窗口
       var adwin_obj=window.open("","adwin","status=no,menubar=no,width=400,
       height=100");
     if(adwin_obj!=null) {
        adwin_obj.document.write("<H1 align=center>优惠促销正在进行中</h1>");
     }
   //-->
   </SCRIPT>
   </BODY>
   </HTML>
```

在浏览器中打开文件，结果如图 3.16 所示，打开的新窗口如图 3.17 所示。

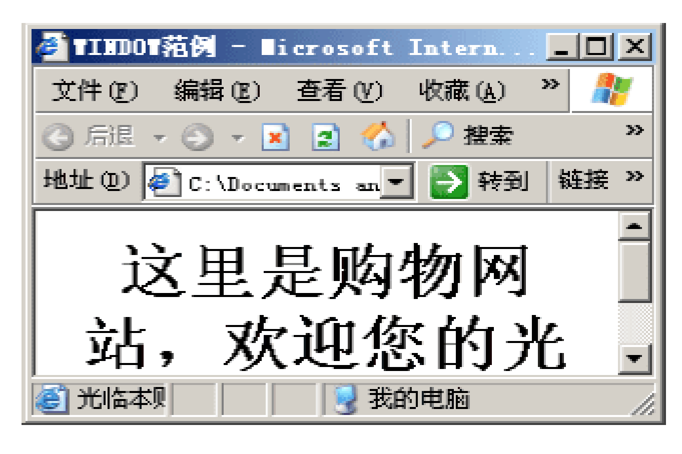


图 3.16　带滚动文字的窗口

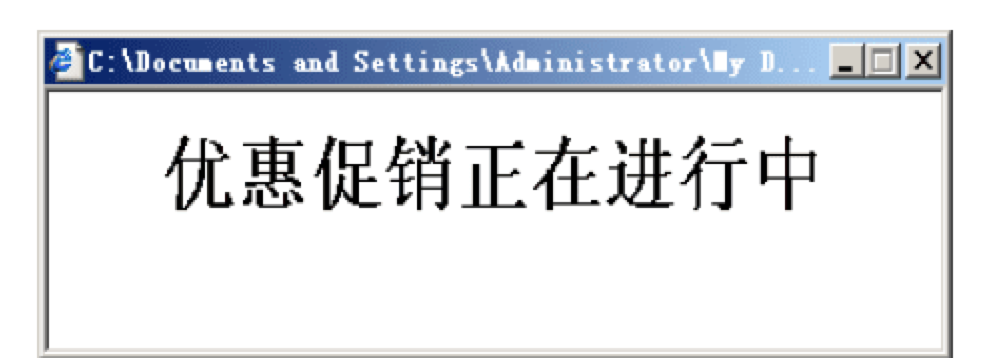


图 3.17　打开的新窗口

3.7.3　验证输入数据的有效性

JavaScript 的最大特点之一是它能够处理表单，验证用户输入数据的有效性。这时的有效性检查在运行浏览器的客户端完成，具有速度快、效率高的特点，如检测输入域是否为空。

【例 3.20】 编写一个网页文件 ex3_20.htm，在网页开始时显示一个提示输入框，要求用户输入一个电子邮件地址。然后检查该地址中是否含有字符@。如果有，便认为这是一个合法的邮件地址，并将其在网页中输出；否则，在网页中输出“非法电子邮件地址”。

网页文件的内容如下：

```
<HTML>
<HEAD>
<TITLE>判定邮件地址的合法性</TITLE>
</HEAD>
<BODY>
<font size=5>
<SCRIPT LANGUAGE="JavaScript">
<!--
   var mstr=prompt("请输入一个电子邮件地址"," ")
   if (mstr.indexOf("@")>0) {
```

```
        document.write(mstr)
    }
    else{
        document.write("非法的电子邮件地址")
    }
//-->
</SCRIPT>
</font>
</BODY>
</HTML>
```

在该例子中，用户输入了电子邮件地址后，利用了 String 对象的 indexOf()函数判定输入的地址中是否包含字符@，然后根据判定的结果输出指定的内容。

3.7.4 验证单选按钮、复选按钮和选择列表值的有效性

单选按钮用于输入性别、出生地等可以从一组信息中选择一项信息的情形。用户在一组单选按钮中作出选择后，程序如何获取用户作出的选择呢？解决的方法是：对整个组的每个按钮逐个检查其 checked 属性，如果某个按钮的 checked 属性值为 true，则说明该项被选中，不用检查后面的按钮的 checked 属性值。

复选按钮在表单中通常用于选择输入人们感兴趣的、喜欢的书籍和电影等信息。当用户在某组信息中选择了若干项后，程序如何获取用户作出的选择呢？解决的方法是：对整个组的每个标记逐一检查其 checked 属性，如果其值为 true,则该项被选中；否则，就没有被选择。

选择列表是为了方便用户输入的表单元素。与单选按钮和复选按钮相比，选择列表在可以提供较多选项的同时只需要占用较少的屏幕空间，这是它的优点。为了获取用户在选择列表中的选择，可以采用类似检查单选按钮和复选按钮的方法，只是这里要检查的是列表中每一项的 selected 属性。如果取值为 true，则该项被选中；否则，没有选中。

由于单选按钮、复选按钮和选择列表检查值的有效性的方法类似。为了节省篇幅，下面仅以选择列表为例说明。

【例 3.21】 编写一个网页文件 ex3_21.htm，让用户利用列表选择输入其喜欢的水果。单击“提交”按钮后，用消息框显示用户的选择。

网页文件的内容如下：

```
<HEAD>
<TITLE>获取选项列表中的信息</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!--
    //获得选择的水果,并提示要求验证
function Results(fruits){
     var yourChoices="";
     for(i=0;i<fruits.length;i++){
       if(fruits[i].selected==true)
```

```
            yourChoices=yourChoices+fruits[i].value+",";
        }
        alert("你的选择是: "+yourChoices); //输出您的选择,要求确认
      }
//-->
</SCRIPT>
</HEAD>
<BODY>
<h1>你喜爱的水果</h1>
<FORM METHOD=POST ACTION=""  name="info">
  <SELECT multiple size="4" name="selects">
      <OPTION value="苹果">苹果</OPTION>
      <OPTION selected value="香蕉">香蕉</OPTION>
      <OPTION value="雪梨">雪梨</OPTION>
      <OPTION value="菠萝">菠萝</OPTION>
      <OPTION value="西瓜">西瓜</OPTION>
      <OPTION value="新奇士橙">新奇士橙</OPTION>
      <OPTION value="新疆哈密瓜">新疆哈密瓜</OPTION>
   </SELECT>
   <br>
<INPUT TYPE="button" value="提交" onClick="Results(document.info.
selects)">
</FORM>
</BODY>
</HTML>
```

在浏览器中打开文件，结果如图 3.18 所示，选择并“提交”后，打开的新的消息如图 3.19 所示。



图 3.18　在列表中选择并提交



图 3.19　确认输入的信息

实　验　3

1．编写一个网页文件，利用 JavaScript 的循环语句和 switch 语句输出 9 行文字：欢迎关临我们学校的网站。要求每 3 行分别使用红、绿、蓝 3 种颜色显示（即 1、2、3 红、绿、蓝；4、5、6 红、绿、蓝；7、8、9 红、绿、蓝）。

2．编写一个网页文件，使用 JavaScript 编写一个 showTitle 函数，该函数可以用指定的字体将指定的内容以 4 号字、居中对齐输出。然后调用 3 次该函数，分别以黑体、隶书和楷体输出标题：计算机与信息管理系。

3．编写一个网页文件。要求，在其中使用 H1 标记显示标题：这是一个有感觉的标题。在浏览器中单击该标题时会显示一个消息框，其中显示文字：你击中我啦!

4．编写一个网页文件。要求，在其中使用 H1 标记显示标题：没有老鼠的感觉真好。在浏览器中把鼠标指向该标题时，其中的文字变为老鼠来啦，赶紧跑呀!当鼠标移开标题后文字恢复原样。

5．编写一个网页文件。要求：其中以标题的形式轮换显示若干个公司的广告语。每个公司的广告语停留 2 秒钟。当显示到某公司的广告语时，单击该标题就可以转向该公司的主页。

6．编写一个网页文件，在网页中显示一个月历。要求：当月的天数必须自动计算得到；在月历中当天的日期用蓝色粗体字显示；单击当天日期则显示一个消息框，其内容为“愿您拥有一个美好的日子”。

7．编写一个网页文件。要求：在打开一个网页时弹出一个没有菜单栏、工具栏和状态栏，宽度和高度分别为 400 和 300 的广告窗口；其中的广告词为“欢迎您的光临!”

8．编写一个网页文件，使用一个表单让用户填写购货订单。填写的信息包括姓名、电话、商品名称、单价、数量和金额。当提交表单时,要求：

（1）商品名称和单价只能让用户选择；

（2）数量为 0 时不予提交；

（3）金额在提交时自动计算，并与所填的“金额”比较；

（4）如（2）、（3）有错误时，则返回已填购货单，并提示错误位置和原因；如果没有错误，则返回已成功提交的页面。

习　题　3

选择题

1．JavaScript 数据类型的表示可以使用（　　）。（多选）

A．二进制　B．五进制　C．八进制　D．十进制　E．十六进制

2．在 JavaScript 中定义变量时要使用（　　）关键字。

A．var　B．dim　C．def　D．function

3．根据 JavaScript 中定义变量的规则，下面描述中正确的是（　　）。

A．变量 dog 和 Dog 是相同的　B．变量 PEOPLE 和 people 是相同的

C．变量 apple 和 Apple 是相同的　D．变量 apple 和 apple 是相同的

4．表达式“55&18”的运算结果是（　　）。

A．55　B．18　C．73　D．65

5．在 JavaScript 中定义一个函数，第一个需要使用的关键字是（　　）。

A．function　B．Function　C．FUNCTION　D．func

6．在下面描述的各种功能中，其中最适合使用自定义函数来实现的是（　　）。

A．在网页中一次性输出 0～100 之间的所有奇数

B．在网页中的多个位置需要显示不同字体、颜色和大小的标题
C．在网页的开头位置处显示一次当天的日期
D．在网页中重复输出 0～100 之间的所有偶数

7．Mousemove 事件发生的条件是（　　）。
A．鼠标在对象表面　　B．鼠标在对象表面上移动
C．鼠标离开了对象表面　　D．鼠标键释放

8．在引用对象属性时，下面的语法格式中正确的是（　　）。
A．对象名->属性　　B．对象名_属性
C．对象名.属性　　D．对象名（属性）

9．为了使一个 HTML 标记作为一个对象响应发生的事件，需要在标记中以（　　）描述的语法格式编写相关的代码。
A．onEvent 事件名="事件处理代码"
B．onEvent 事件名=事件处理代码
C．on.事件名="事件处理代码"
D．on 事件名="事件处理代码"

10．为了对网页中的标记对象进行操作，需要设置标记的（　　）属性。
A．ID　　B．SRC　　C．VALUE　　D．DEFAULT

11．在 JavaScript 中要建立一个内置对象时，一般使用（　　）关键字。
A．create　　B．new　　C．build　　D．open

12．为了使字符串对象输出一个全小写字母输出的结果，可以使用字符串对象的（　　）方法。
A．toLowercase　　B．toLowerCase
C．ToLowerCase　　D．ToLowercase

13．在 JavaScript 中，一个字符串对象的长度是指该字符串包含（　　）的个数。
A．字符　　B．字节　　C．字长　　D．字符或字节

14．下面关于日期对象的描述中，正确的是（　　）。
A．日期对象只提供对日期信息的管理操作
B．日期对象只提供对时间信息的管理操作
C．日期对象提供对日期信息和时间信息的管理操作
D．日期对象提供对日期信息的管理操作，时间对象提供对时间信息的管理操作

15．为了在浏览器窗口的状态栏中设置指定的字符串内容，需要设置 window 对象的（　　）属性。
A．status　　B．statusText
C．statusBar　　D．statusValue

16．在一个网页文件中如果需要打开一个新的窗口显示内容，则可以使用 window 对象的（　　）方法。
A．new　　B．open　　C．alert　　D．prompt

17．利用下面的（　　）语句可以使网页自动转向指定的 URL。
A．location.hash= http://www.163.com

B．location.URL= "www.163.com "

C．location.href="http://www.163.com"

D．location.href="www.163.com "

18．为了在表单提交前检查表单数据的有效性，以便决定是否提交表单到服务器，可以通过编写响应（　　）事件的代码来实现。

A．click　　B．mouseDown　　C．mouseUp　　D．submit

19．为了方便数据处理，表单中同一组的单选项目其名称通常是（　　）。

A．不同的　　B．无关紧要　　C．相同的

20．下面哪一个不是 JavaScript 的数据类型？（　　）

A．String　　B．Date　　C．Boolean

D．Object　　E．Intergers

第 4 章　构建基于 PHP 5 的动态 Web 开发环境

本章导读

PHP 是一种跨平台的服务器端的嵌入式脚本语言。它大量的借用 C、Java 和 Perl 语言的语法，又突出了 PHP 自身的特征，使 Web 开发者能够快速地写出动态网页。PHP 5 是 PHP 4 的发展，新增了许多功能。PHP 程序是运行在服务器端的程序，因此需要首先搭建一个 PHP 程序能够运行的平台。

本章将简单介绍 PHP 5 的发展历程、特点、新增的功能，使读者对 PHP 语言和 PHP 5 版本有一个大致的认识。接着主要介绍 Windows 和 Linux 平台下如何构建基于 PHP 5 的动态 Web 开发环境，使读者熟悉在 Windows 或 Linux 操作系统的不同环境下如何搭建一个有效的平台。

4.1　PHP 5 概述

PHP 是一种服务器端的脚本/编程语言，是一种简单的、面向对象的、解释型的、健壮的、安全的、性能非常高的、独立于架构的、可移植的、动态的脚本语言。PHP 以方便快速的风格迅速在 Web 系统开发中占有了重要地位，PHP 本身提供了大量的函数。PHP 作为开放源代码脚本语言，正成为世界上最流行的 Web 应用程序编程语言之一。

PHP 最初是由 Rasmus Lerdorf 在 1994 年创建的。它开始是一个简单地用 Perl 语言编写的程序，是用来记录 Rasmus Lerdorf 本人在线简历的访问者的，以后它又被用 C 语言重写了一遍，范围扩大到访问数据库。在 1995 年以 Personal Home Page Tool（PHP tool）为名开始对外发表第一个版本。在这个早期版本中，提供了访客留言本、访客计数器等简单的功能。1995 年中，第二版的 PHP 问市，定名为 PHP/FI（form interpreter）。在其中加入了 mSQL 的支持，自此奠定了 PHP 在动态网页开发上的影响力。在 1996 年底，有 15 000 个 Web 网站使用 PHP/FI；在 1997 年中，使用 PHP/FI 的 Web 网站增长到超过 50 000 个。而在 1997 年中，开始了第三版的开发计划，开发小组加入了 Zeev Suraski 及 Andi Gutmans 两人，第三版定名为 PHP 3。PHP 3 跟 Apache 服务器紧密结合的特性，加上它不断地更新及加入新的功能，并且几乎支持所有主流与非主流数据库，再以它高速的执行效率，使得 PHP 在 1998 年末，PHP 的安装人数几近 10 000，有大约 100 000 个网站报告他们使用了 PHP，1999 年中使用网站超过了 150 000 个！2000 年 5 月，PHP 4.0 正式发布，它使用了 Zend 网页内嵌程序引擎，提供更高的性能，还包含了其他一些关键功能，例如，支持更多的 Web 服务器，HTTP Sessions 支持，输出缓存（output buffering），更安全的处理用户输入的方法以及一些新的语言结构。PHP 4.0 是更有效的，更可靠的动态 Web 开发工具，在

大多数情况运行比 PHP 3.0 快，其脚本描述更强大并且更复杂，最显著的特征是速率的增加。2004 年 7 月 PHP 5 问世，无论对于 PHP 语言本身还是 PHP 的用户来讲，PHP 5 发布都算得上是一个里程碑式的版本。PHP 5 的诞生，使 PHP 编程进入了一个新时代。Zend II 引擎的采用，完备对象模型、改进的语法设计，终使得 PHP 成为一个设计完备、真正具有面向对象能力的脚本语言。

使用 PHP 编程的最大好处是学习这种编程语言非常容易，而且 PHP 含有许多库，即使对需要使用的函数不是十分了解，也能够概略地知道如何完成一个特定的任务。PHP 是一种易于学习和使用的服务器端脚本语言。只需要很少的编程知识就能使用 PHP 建立一个真正交互的 Web 站点，PHP 是能让你生成动态网页的工具之一。PHP 网页文件被当作一般 HTML 网页文件来处理，并且在编辑时可以用编辑 HTML 的常规方法编写 PHP。

PHP 是完全免费的，可以从 PHP 官方站点（http://www.php.net）自由下载。PHP 遵守 GNU 公共许可，在这一许可下诞生了许多流行的软件诸如 Linux 和 Emacs。可以不受限制的获得源码，甚至可以从中加进自己需要的特色。PHP 在大多数 UNIX 平台、GUN/Linux 和微软 Windows 平台上均可以运行。怎样在 Windows 环境或 UNIX 环境下安装 PHP 的资料可以在 PHP 官方站点上找到，而且安装过程也很简单。

PHP 正迅速变成一种标准的，多用途的，面向对象的脚本语言，成为比以前更有吸引力的一种技术。在这里不是就 PHP 的技术或技巧进行深入探讨，而是重在介绍，相信通过学习会有更多的人喜欢上 PHP 语言，这样我们的目的也就达到了。

PHP 作为风靡全球的网站后台开发语言，拥有许多个性鲜明的特点，下面就其基本特性作一介绍。

- 开放源码——所有的 PHP 源码可以从网上得到。
- 免费使用——PHP 是一个免费软件，尽可放心使用。
- 基于服务器端——由于 PHP 是在 Web 服务器端运行的，所以它的程序可以很大、很复杂而不会降低客户端的运行速度。
- 跨平台——PHP 程序可以在 UNIX、Linux 或者 Windows 操作系统下运行。
- 嵌入 HTML——PHP 程序可以嵌入到 HTML 内部。
- 语言简单——与 Java 和 C++不同，PHP 语言坚持以基本语言为基础，语法简单。
- 效率高——和其他解释性语言相比，PHP 系统消耗较少的系统资源。
- 分析 XML——用户可以组建一个可以读取 XML 信息的 PHP 版本。
- 数据库模块——用户可以使用 PHP 存取 Oracle、Sybase、MS SQL、MySQL、mSQL、dBase 等以及任何支持 ODBC 标准的数据库。
- 文件存取——PHP 有许多支持文件存取的函数。
- 文本处理——PHP 有许多函数处理字符串，其中包括模式匹配的能力。
- 复杂的变量——PHP 支持标量、数组等变量，这给用户提供了支持其他高级数据结构的坚实基础。
- 图像处理——用户可以使用 PHP 动态地创建图像。

PHP 5 比前面两个版本 PHP 3、PHP 4 有许多成功的地方，新的特点和改变去掉了以前 PHP 版本缺点。下面将对 PHP 5 及它的新特征进行详细的描述。

1．新的面向对象模型

在 PHP 3 中首先引入了面向对象模型，在 PHP 4 中重写了脚本引擎，这使得程序运行得更快，更稳定；但对象模型有一些限制，这使得 PHP 5 版本成为值得关注的焦点。例如在 PHP 5 以前，当一个指向特定对象的变量赋值给另一个变量时，将产生这个对象的副本，这两个变量不是指向同一个对象，导致改变一个对象不会改变相应的对象。PHP 5 的对象模型把对象看成与任何其他数据类型不同，通过引用来传递。PHP 不要求通过引用（reference）显式传递和返回对象，它是 PHP 5 中最重要的新特性。在 PHP 的前几个版本中，脚本默认复制对象，现在 PHP 5 只移动句柄，需要更少的时间。脚本执行效率的提升是由于避免了不必要的复制。在对象体系带来复杂性的同时，也带来了执行效率上的收益。同时，减少复制意味着占用更少的内存，可以留出更多内存给其他操作，这也使效率提高。

2．新的内存管理器

PHP 5 使用了 Zend 引擎Ⅱ，对象被存储于独立的结构 Object Store 中，而不像其他一般变量那样存储于 Zval 中（在 PHP 4 中对象和一般变量一样存储于 Zval）。在 Zval 中仅存储对象的指针而不是内容（value）。当人们复制一个对象或者将一个对象当作参数传递给一个函数时，不需要复制数据，仅仅保持相同的对象指针并由另一个 Zval 通知现在这个特定的对象指向的 Object Store。由于对象本身位于 Object Store，人们对它所做的任何改变将影响到所有持有该对象指针的 Zval 结构，表现在程序中就是目标对象的任何改变都会影响到源对象。这使 PHP 对象看起来就像总是通过引用（reference）来传递，因此 PHP 中对象默认为通过“引用”传递，不再需要像在 PHP 4 中那样使用&来声明。

某些语言，最典型的如 C，需要显式地要求分配内存来创建数据结构。一旦分配到内存，就可以在变量中存储信息。同时也需要在结束时释放内存，这使机器可以空出内存给其他变量，避免耗光内存。PHP 可以自动进行内存管理，清除不再需要的对象。PHP 使用了引用计数（reference counting）这种单纯的垃圾回收（garbage collection）机制。每个对象都内含一个引用计数器，每次引用连接到对象，计数器加 1。当引用离开生存空间或被设为 NULL，计数器减 1。当某个对象的引用计数器为零时，PHP 知道用户将不再需要使用这个对象，释放其所占的内存空间。

3．使用统一的构造器

代替用类名作为构造器，在 PHP 5 中使用统一的构造器（_ _construct()）来构造对象，这使书写构造方法时更方便。

4．通过定义“_ _destructor()”方法支持对象析构器

在 PHP 5 中，当一个对象不需要时，允许定义并使用析构器函数“_ _destructor()”。

5．异常处理

在 PHP 5 中增加了使用 try/throw/catch 进行异常处理功能。

6．XML and Web Services

在 PHP 5 中 XML 的更新可能是最有意义的事。在 PHP 5 中改善的 XML 函数功能使 PHP 技术在一些领域可以与别的 Web 技术媲美，甚至在一些方面超过别的 Web 技术。

7．不再支持 Windows 95

2002 年，微软正式宣布停止支持 Windows 95，PHP 开发组织明智的决定不再支持 Windows 95。因此 Windows 95 不再保证 PHP 的运行和实现 PHP 使用的功能。

8．PHP 5 的访问方式允许限制对类成员的访问

这是在 PHP 5 中新增的功能，像 C++和 Java 一样，PHP 有 public、private 和 protected 三种访问方式。有了访问方式，才能开发一个可靠的面向对象应用程序，并且构建可重用的面向对象类库。

9．新增的其他一些语言特性

1）类型指示

PHP 是一种弱类型的语言，也就是说，在使用变量前不需要定义，不需要声明变量的数据类型。这给编程带来很多便利，但也带来一些隐患，特别当变量的类型变化时。在 PHP 5 增加了类型指示，可以在执行过程中自动对类方法的参数类型进行判断。在强类型语言中，所有变量的类型将在编译时进行检查，而 PHP 在运行时使用类型指示来对类型进行检查。如果类方法参数的类型不对，将会报出类似“Fatal error: Argument 1 must implement interface Bar…”这样的错误信息。

2）final 关键字

PHP 5 中新增加了 final 关键字，它可以加在类或类方法前。标识为 final 的类方法，在子类中不能被重写。标识为 final 的类，不能被继承，而且其中的方法都默认为 final 类型。

3）对象复制

前面在内存管理部分说过，PHP 5 中默认通过引用传递对象。像使用$object2=$object1 这样的方法复制出的对象是相互关联的。如果确实需要复制出一个值与原来相同的对象而希望目标对象与源对象没有关联（像普通变量那样通过值来传递），那么就需要使用 clone 关键字。如果还希望在复制的同时变动源对象中的某些部分，可以在类中定义一个__clone()函数，加入相应操作。

4）类常量

PHP 5 中可以使用 const 关键字来定义类常量。

5）__METHOD__常量

__METHOD__是 PHP 5 中新增的“魔术”常量，表示类方法的名称。魔术常量是一种 PHP 预定义常量，它的值可以是变化的，PHP 中的其他已经存在的魔术常量有__LINE__、__FILE__、__FUNCTION__、__CLASS__等。

总之，PHP 5 中已在 PHP 4 的基础上做了大量的改进，前面提到的仅是主要的改进。其他改进包括其他特征、许多修补的 bug、性能大大提高的框架等。

4.2 构建 Windows 的动态 Web 服务器

在安装 PHP 作为 WWW 服务器的一部分时，可以考虑 UNIX 系统或 Windows NT /2000/XP/2003 等 Windows 平台。在 PHP 发展过程中，开始只支持基于 UNIX 体系和类 UNIX 体系的操作系统，如 Linux 和 SCO UNIX。但是随着 PHP 的发展，其使用也逐渐增多，人们觉得缺少集成环境的 Linux 下进行程序的开发是不理想的，所以在 Windows 下的 PHP 版本很快被开发出来。一般来说，除非实在没有条件构建 UNIX 平台时，才在 Windows 下进行程序的开发和调试。但是，在目前 Linux 应用非常广泛的情况下，流行的开发方式是

在一台计算机上搭建Linux平台作为服务器，另一台安装Windows平台的计算机通过Telnet来对 Linux 进行管理。所以，PHP 程序开发工作应该从 Windows 下开始，源码编写应在Windows 下进行，然后通过 Telnet 在 Linux 上进行最后的调试和试运行。这样配合才可以使工作效率达到最高。

Linux 系统方面，可以选择各式的 Linux 套件，包括 Slackware Linux、RedHat、OpenLinux、SUSE 等，这方面的软件可以去网站下载完整的系统进行安装。

Apache 服务器则是目前网站采用最多的服务器，可以到 http://www.apache.org 下载最新版的程序及相关文件。

PHP 则可以去它的官方网站 http://www.php.net 下载所需要的程序。虽然目前 Windows 2000 或者 Windows 2003 等平台也能安装 PHP 及 Apache 服务器，不过 PHP 和 Apache 在Linux 下运行得更快更好。

当然，若想使用商业化的作业平台，SUN、IBM、HP、DEC、SGI、NEC 等公司都有提供相关的 UNIX 或者是 WIN NT 的作业平台。加上高安全性调整过后的 Apache 服务器：Stronghold 或其他支援 SSL 的 Apache 版本。这种组合，相信能满足商业化的需求，而 PHP就扮演着快速方便的 CGI 角色，让客户对站点的服务品质更加满意。

对于一个程序员来说，一个有效的设计调试环境往往能大大减轻程序设计的劳动量，而在今天，虽然程序的网络化，Web 化已是不可避免的潮流，但在 Web 程序开发上却没有一个“以人为本”的像 VB、JBuilder 那样强大的集成开发环境。这主要还是因为 Web 程序开发还是一个发展中的事物，还没一个标准可循，所以有时不得不在多种开发环境、多种系统、多种工具之间切换，因此如何设置一个尽可能多种多样又合理有效的开发环境是进行 Web 程序设计一个不可小看的问题。下面将详细描述在 Windows 下如何构建动态的Web 服务器。

4.2.1 IIS 和 PHP 5 的组合安装和测试

虽然大家都了解 Linux，但单纯从使用角度来说还是习惯于 Windows，因为 Windows的确有它的特点，尤其是 IIS，既不失功能强大，又易于管理和配置。所以即使是在 UNIX类服务器占市场主流的今天，还是在 Windows 下调试 Web 程序更方便更有效。使用Windows 系统的最大好处是可以直接享用 Windows 下友好的集成环境。在客户端浏览器方面，这是 Windows 的强项，只需使用它提供的浏览器 IE 就可以获得比较好的效果。

如果 IIS 是服务器类操作系统默认的组件，则在安装时自动安装，其后不需要再装；否则，在安装操作系统时可以同时安装，也可以在操作系统安装完成后单独安装。安装过程比较简单，由于篇幅有限，在此不作解释，后面将以 Windows 2003 为例说明如何安装和配置 IIS。下面将学习如何安装 PHP。

下面安装 PHP 的方法适用于 Windows 98/ME 以及 Windows NT/2000/XP/2003 等Windows 平台。PHP 不能在 16 位平台例如 Windows 3.1 下运行。有时把支持 PHP 的 Windows平台称为 Win32。自 PHP 4.3.0 开始不再支持 Windows 95。

在 Windows 平台下有两种方法安装 PHP，即安装程序安装和手工安装。

1．安装程序安装

Windows 版的 PHP 安装程序可以从 http://www.php.net 获得，此程序将安装 PHP 的

CGI 版本，并且自动配置好 IIS Web 服务器。

尽管向导安装程序是一种使 PHP 运行起来的简单方法，但是有很多限制，例如，不支持自动设置扩展库。只有下载 zip 压缩包才包含所有支持的扩展库，因此使用安装程序安装 PHP 不是最好的方式。

首先，在系统中安装自己选择的 HTTP（Web）服务器，并确认它正常工作。 然后运行可执行的安装程序并按照安装向导的提示进行安装。安装程序支持两种安装方法：标准，将使用合理的默认配置进行安装；高级，在安装过程中提几个问题。安装向导会收集足够的信息来设置 php.ini 文件，并配置好 Web 浏览器以使用 PHP。

一旦安装完成，安装程序会提示重新启动系统、重启服务器或直接开始使用 PHP。值得一提的是这样安装的 PHP 并不安全。如果想要更安全的安装 PHP，最好手工安装，并且小心地配置每个选项。自动进行的安装程序只是提供一个可以马上使用的 PHP，并不意味着可以用于在线的服务器上。

2．下载安装

首先，需要从 http://www.php.net/downloads.php 下载一个包含可执行版本的 zip 包。

不论何种 Web 服务器，都需要先进行以下步骤，其中（1）～（5）步是安装和设置 PHP 的过程，（6）～（12）步是在 NT/2000/XP 系统下配置服务器的过程。

（1）将 PHP 压缩包释放到选择的目录中。为方便以及版本无关起见，以下步骤中假定 PHP 位于 C:\php 中。可以选择其他路径但最好不要用中间有空格的路径（例如，C:\Program Files\PHP），如果这样做有些 Web 服务器会崩溃。

（2）无论使用 CGI 接口还是 ISAPI（Internet 服务器应用程序编程接口）都应确保 php5ts.dll 能正确地被搜索到，有下面三个可选择的方法。

第一个选择是复制该文件到 Windows 系统目录，如对于 Windows 9x/ME 的系统目录是 C:\Windows\system，对于 Windows NT/2000 的系统目录是 C:\winnt\system32，对于 Windows NT/2000 服务器版的系统目录是 C:\winnt40\system32，对于 Windows XP/2003 的系统目录是 C:\Windows\system32。

第二个选择是复制该文件到 Web 服务器的目录，如 IIS 的 Web 服务器的目录是 C:\Inetpub\wwwroot，apache 的 Web 服务器的目录是 C:\Program Files\Apache Group\Apache2\bin。

第三个选择是把 PHP 目录（例如 C:\php）添加到 PATH 环境变量中。为了将来更好地维护，建议使用最后一个选择，将 PHP 目录添加到 PATH 环境变量中，因为这样更便于将来升级PHP。具体操作如下：进入控制面板并打开“系统”图标（“开始”→“设置”→“控制面板”→“系统”，Windows XP/2003 中是：“开始”→“控制面板”→“系统”）；选择“高级”标签页；单击“环境变量”按钮；在“系统变量”栏中，找到 Path 这一项，鼠标双击 Path 这一项；在最后加入 PHP 目录，包括前面的“;”（例如；C:\php），单击“确定”按钮并重新启动计算机。

（3）为 PHP 设定有效的配置文件：php.ini。压缩包中包括两个 ini 文件，php.ini-dist 和 php.ini-recommended。建议使用 php.ini-recommended，因为在该文件中优化了性能和安全。仔细阅读此文件中的说明并研究 ini 设置来人工设定每个项目。尽管 PHP 在默认的 ini 文件下也是安全的，但是如果要达到最佳的安全效果，则最好使用 php.ini-recommended 文件。

将选择的 ini 文件复制到 PHP 能够找到的目录下并改名为 php.ini。

php.ini 默认的搜索路径的顺序是：SAPI 模块所指定的位置，HKEY_LOCAL_MACHINE\SOFTWARE\PHP\IniFilePath（Windows 注册表位置），PHPRC 环境变量，当前工作目录，Web 服务器目录（对于 SAPI 模块）或 PHP 所在目录（Windows 下除 SAPI 外的其他情况），Windows 目录（C:\Windows 或 C:\winnt）或--with-config-file-path 编译时选项指定的位置。

注意： *如果在 Windows NT/2000/XP/2003 中使用 NTFS 分区，请确认运行 Web 服务器的用户有权限读取 php.ini（例如，将其设置为 Everyone 可读）。*

以下步骤为可选项。

（4）编辑 php.ini 文件，将 doc_root 指向 Web 服务器的主目录。

```
doc_root = c:\inetpub\wwwroot            //对于 IIS/PWS
doc_root = c:\apache\htdocs              // 对于 Apache
```

（5）在 Windows 下安装完 PHP 和 Web 服务器之后，可能想要安装一些扩展库来获得更多功能。可以通过修改 php.ini 来选择当 PHP 启动时加载哪些扩展库。如 extension_dir = C:\php\extensions，其中“C:\php\extensions”表示扩展库所在的目录。也可以在脚本中通过使用 dl( )来动态加载。

注意： *在新安装之后建议先确定 PHP 在没有任何扩展时运行正常，然后再在 php.ini 中加载扩展库。*

PHP 现在已经安装在系统中了，下一步是为 Web 服务器启动 PHP，也就是配置服务器。在 Windows NT/2000/XP 中的 IIS 4 或更新版本中安装 PHP 后配置服务器 IIS，按照以下说明进行。

（6）有两种选择安装 PHP：CGI 方式（即安装程序安装）或者 ISAPI 模块方式（手工安装）。无论哪种方式，都需要打开管理控制台（如 Windows 2000/XP 中的控制面板→管理工具）。用右键单击 Web 服务器节点（多数是“默认站点”），选择“属性”。

如果选择 CGI 方式，请参照（7）～（9）三个步骤。

（7）选择“主目录”、“虚拟目录”或者“目录”标签栏，将执行权限改为“纯脚本”；单击“配置”按钮，然后选择“应用程序映射”标签栏。

（8）单击“添加”按钮，在“可执行文件”框中输入 c:\php\php-cgi.exe（假定 PHP 被解压缩到 c:\php\）。

（9）在“扩展名”框中输入要使用的 PHP 后缀（.php）。“动作限制”为空，选中“脚本引擎”。也可以选中“检查文件是否存在”——IIS 会先检查脚本文件是否存在，导致略微损失一点性能。这样当出现错误时会输出错误信息而不是 PHP 没有输出任何数据的 CGI 错误。然后单击“确定”按钮。

对于其他想使用的 PHP 后缀重复这个步骤。通常使用的有.php 和.phtml，对于一些旧的程序还需要.php4。

（10）设定合适的目录安全性（在 Internet 信息服务中完成）。如果 NT 服务器使用 NTFS 文件系统，给 I_USR_用户加上 php.exe/php-cgi.exe 文件所在目录的可执行权限（通过资源管理器完成）。

如果选择 ISAPI 模块，请参照（11）、（12）两个步骤。

（11）如果不想用 PHP 进行 HTTP 认证的话，应该跳过这一步。在“ISAPI 筛选器”中，添加一个新的 ISAPI 筛选器。用 PHP 作为“筛选器的名称”，并且把 php5isapi.dll 作为可执行文件的路径。

（12）选择“主目录”、“虚拟目录”或“目录”标签栏，然后将执行权限改为“纯脚本”。

（13）单击“配置”按钮，选择“应用程序映射”标签栏。单击“添加”按钮，将“可执行文件”指向适当的 ISAPI DLL。例如，PHP 5 的值可能是 C:\php\php5isapi.dll。在“扩展名”中填入 php。选择“全部动作”（或者“限制为”留空），选中“脚本引擎”。然后单击“确定”按钮。

（14）停止 IIS 服务，然后重新启动 IIS 服务。

（15）用文本编辑器编写如下语句：

```
<?
phpinfo();
?>
```

保存文件名为“test.php”到 C:\Inetpub\wwwroot 目录，然后打开浏览器浏览 http://localhost/test.php，出现 PHP 基本信息，如图 4.1 所示，就说明配置 PHP 成功。严格按以上说明安装配置，都会一次成功。

Windows 2003 Server 的 IIS 的安装与配置与前面说明的略有不同，下面以 IIS 6（Windows 2003 Server）为例说明在 Windows 下如何安装和配置 IIS。

首先通过“管理您的服务器”窗口中的“添加或删除角色”功能将 IIS 安装好。在 Windows Server 2003 里面，IIS 称为“应用程序服务器”，如图 4.2 所示。安装时如果需要，可以选中 ASP.NET 等选项。不过就算没选中，以后也可以很方便启用。

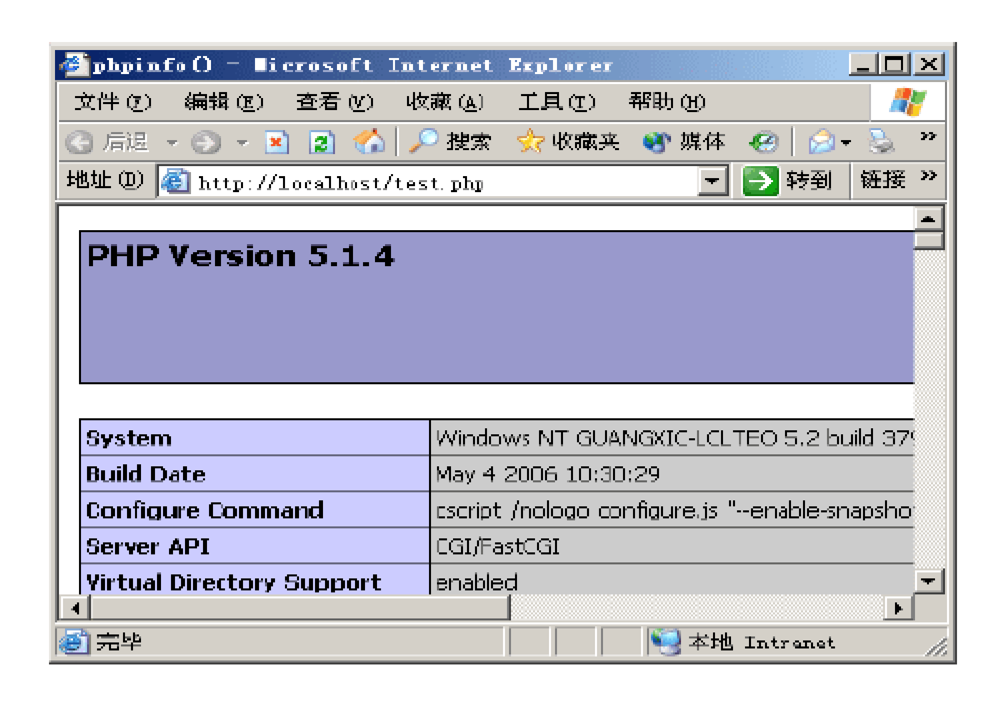


图 4.1　php 5 的基本信息

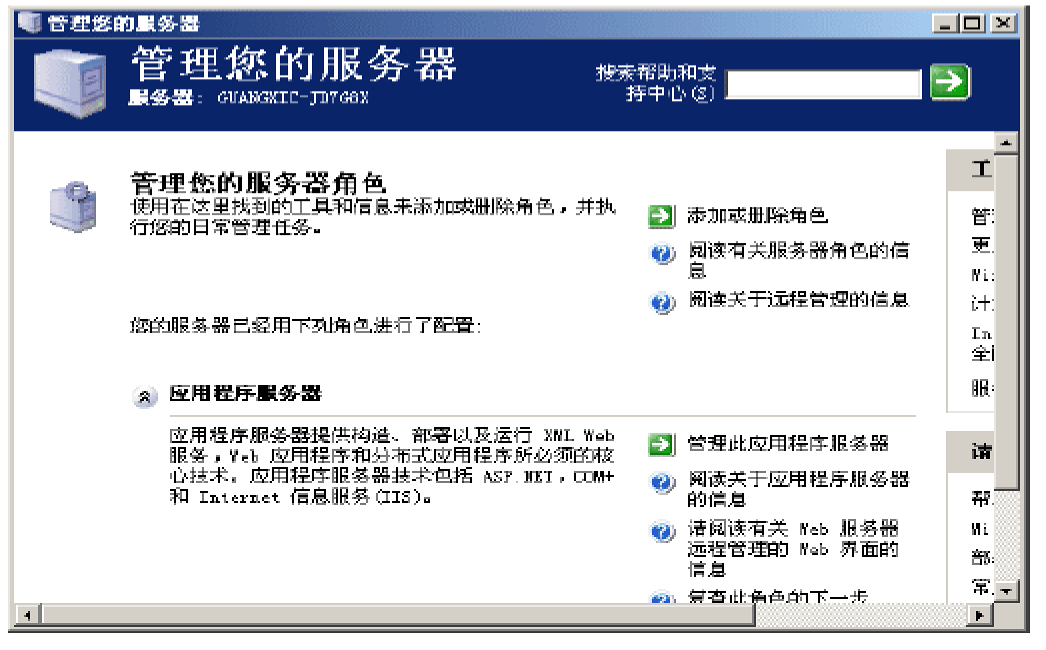


图 4.2　“管理您的服务器”窗口

在“管理您的服务器”窗口中，单击“管理此应用程序服务器”连接，打开“应用程序服务器”管理窗口。然后从左侧依次选中“Internet 信息服务（IIS）管理器”→“本地计算机”→“Web 服务扩展”。接着选中右边任意一个项目，单击“添加一个新的 Web 服务扩展”连接，如图 4.3 所示。

输入扩展名，并单击“添加”按钮将 C:\php\php5isapi.dll 文件添加到列表中，如图 4.4 所示。最后选中“设置扩展状态为允许”复选项，并单击“确定”按钮关闭对话框。现在应该在“Web 服务扩展”列表中可以看到刚刚添加的项目了。

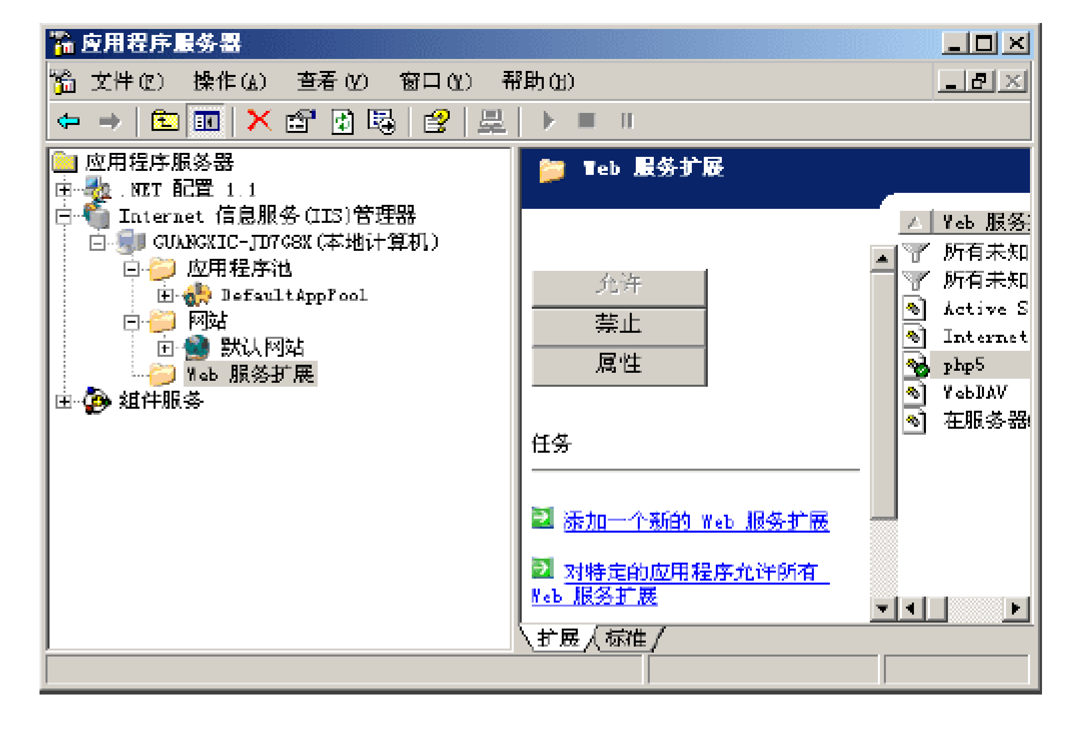


图 4.3 “应用程序服务器”窗口

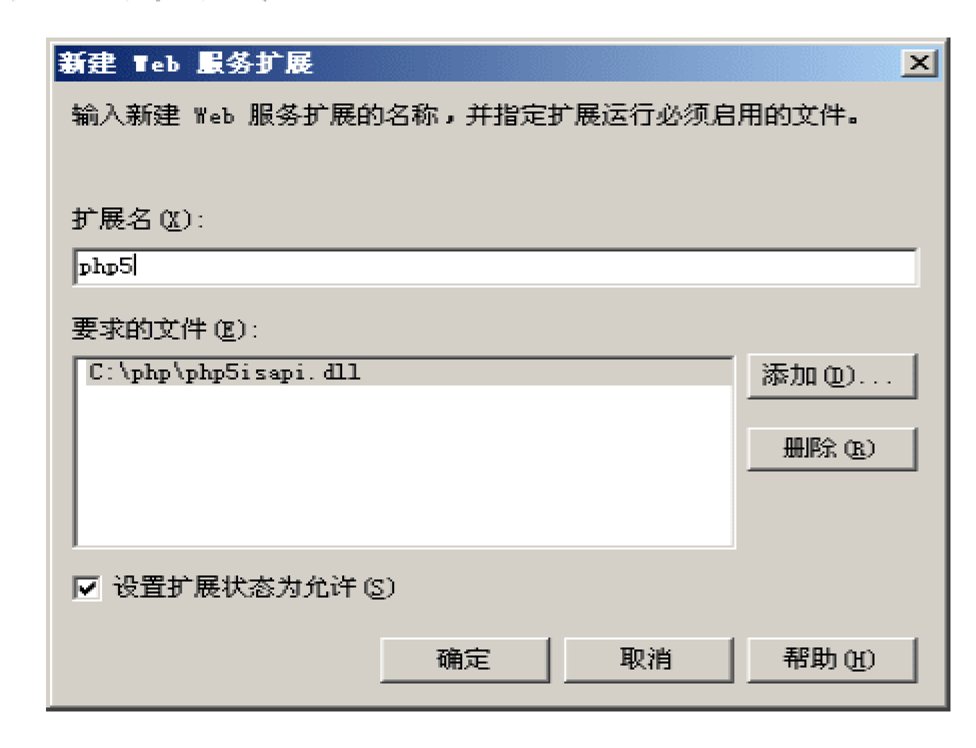


图 4.4 添加 Web 服务扩展

注意：PHP 解析分为 cgi 和 isapi 两种模式，如果此处选择了 php_cgi.exe，那步骤 3 中一定保持一致！

3. 从窗口中安装

在“应用程序服务器”管理窗口左侧依次选中“Internet 信息服务（IIS）管理器”→“网站”。然后在“默认网站”项目上单击鼠标右键选择“属性”，打开“默认网站 属性”对话框。

切换到“主目录”选项卡，单击“配置”按钮，打开“应用程序配置”对话框。再单击“添加”按钮，打开“添加/编辑应用程序扩展名映射”对话框。

单击“浏览”按钮，选中 ISAPI 文件（php5isapi.dll）或者 CGI 文件（php-cgi.exe），并按照图 4.5 中的选项进行设置。最后一直单击“确定”按钮返回“默认网站 属性”对话框。

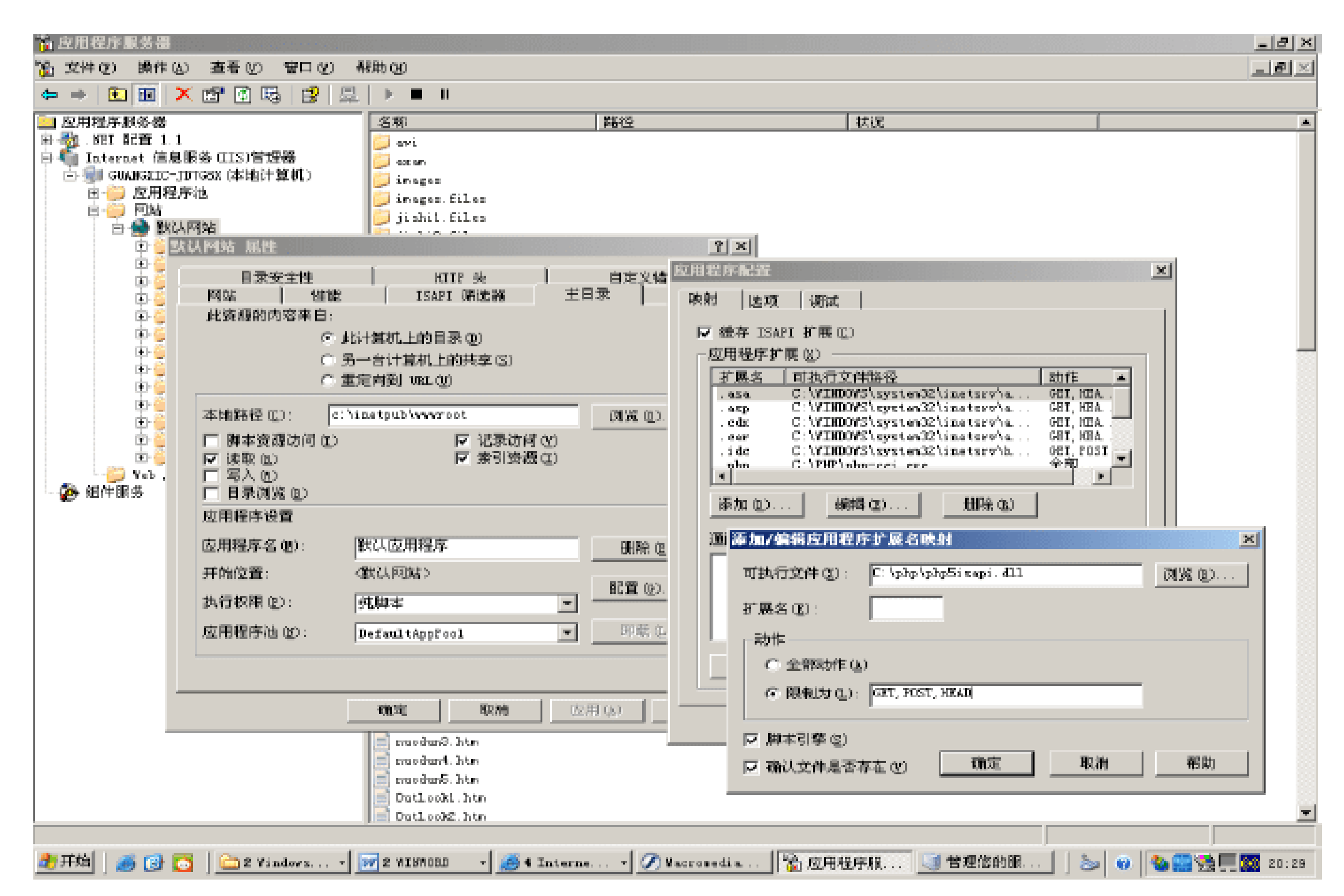


图 4.5 添加应用程序扩展名映射

切换到“文档”选项卡，如图 4.6 所示，单击“添加”按钮，将 index.html 和 index.php 添加到默认内容文档列表中，如图 4.7 所示。最后单击“确定”按钮关闭对话框。

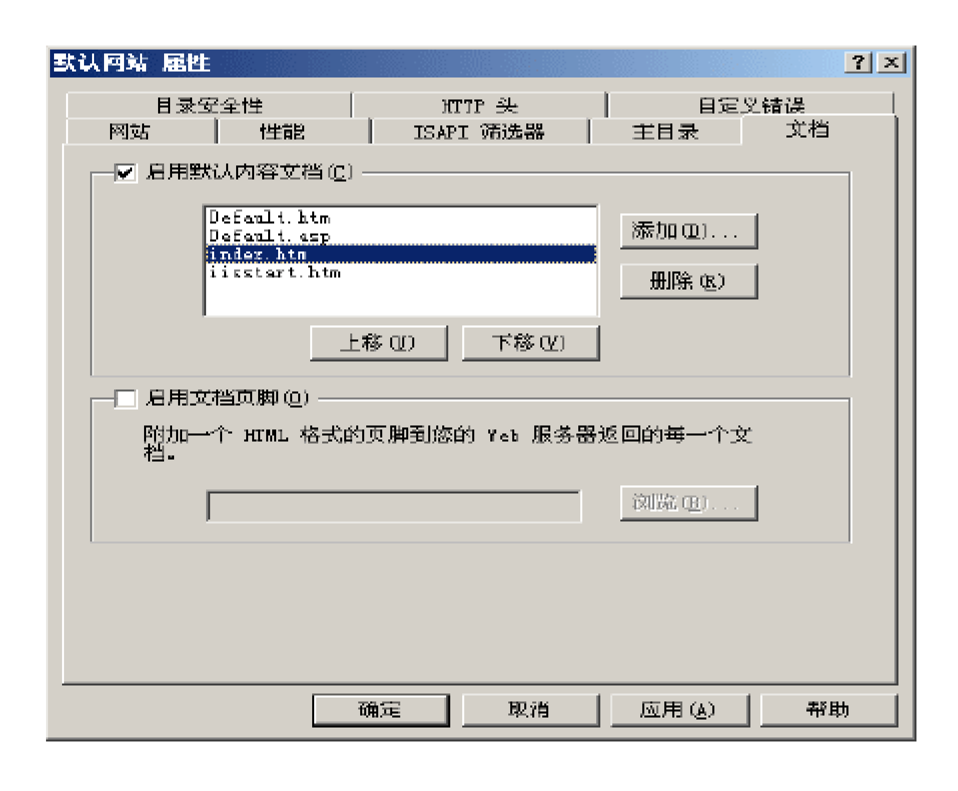




图 4.6 “文档”选项卡



图 4.7 添加默认文档

如果过些时候之后碰到 CPU 占用率达到 100%，则取消选中“缓存 ISAPI 应用程序”（“主目录”下单击“配置”按钮”）。

PHP 与 IIS 组合安装时，应该注意如下的一些事项：

（1）CGI 用户必须在 php.ini 中将 cgi.force_redirect 指令设为 0。此外，CGI 用户可能需要设定 cgi.redirect_status_env 指令。在使用这些指令时，确保它们没有在 php.ini 中被注释。

（2）修改 Windows 的 PATH 环境变量以把 PHP 目录包括进去。这样 PHP 的 DLL 文件、可执行文件和 php.ini 就都可以保留在 PHP 目录中而不用把 Windows 系统目录搞乱了。

（3）IIS 专用用户（通常为 IUSR_MACHINENAME）需要能够读取各个文件和目录的权限，例如，php.ini、docroot 和 session 的 tmp 目录。

（4）确保在 php.ini 中正确设定了 extension_dir 和 doc_root 指令的值。这些指令依赖于 PHP 被安装的系统。在 PHP 5 中 extension_dir 是 ext，因此在 PHP 5 中 extensions_dir 的一个取值例子是“c:\php\ext”，IIS 的 doc_root 的取值例子是“c:\Inetpub\wwwroot”。

（5）PHP 扩展库的 DLL 文件，如 php_mysql.dll 和 php_curl.dll，存放于 PHP 下载的 zip 包中（自动安装包里没有）。在 PHP 5 中，很多扩展库都是 PECL 的一部分，可以从“Collection of PECL modules”包中下载，例如，php_zip.dll 和 php_ssh2.dll。

（6）在定义应用程序扩展名映射时，应选中“检查文件是否存在”。以极小的性能为代价，IIS 会在调用 PHP 之前检查脚本文件是否存在并选出认证方法。这意味着 Web 服务器会提供一个有道理的 404 形式错误信息而不是一条 CGI 错误说什么 PHP 没有输出任何数据。

4.2.2 Apache 和 PHP 5 的组合安装和测试

著名的 Web 服务器 Apache 和现在互联网上应用广泛的数据库 MySQL 的出现，为 Windows 下 PHP 工作环境提供了新的组成方案。

1．Apache 的安装

（1）从 www.apache.org/dist/httpd/binaries/win32 下载 Apache 服务器 Apache 2.0。双击安装文件进行安装。程序默认安装路径为 C:\ program files\apache Group\。

（2）安装完成后，Apache 服务自动加载，这时打开浏览器浏览 http://localhost/，出现 Apache 欢迎页面，如图 4.8 所示，这步需要将 C:\apache2\htdocs 目录中的文件“index.html.zh-cn.gb2312”改为“index.html”，方能显示。如果这步出现异常，请检查安装源文件，重新安装。

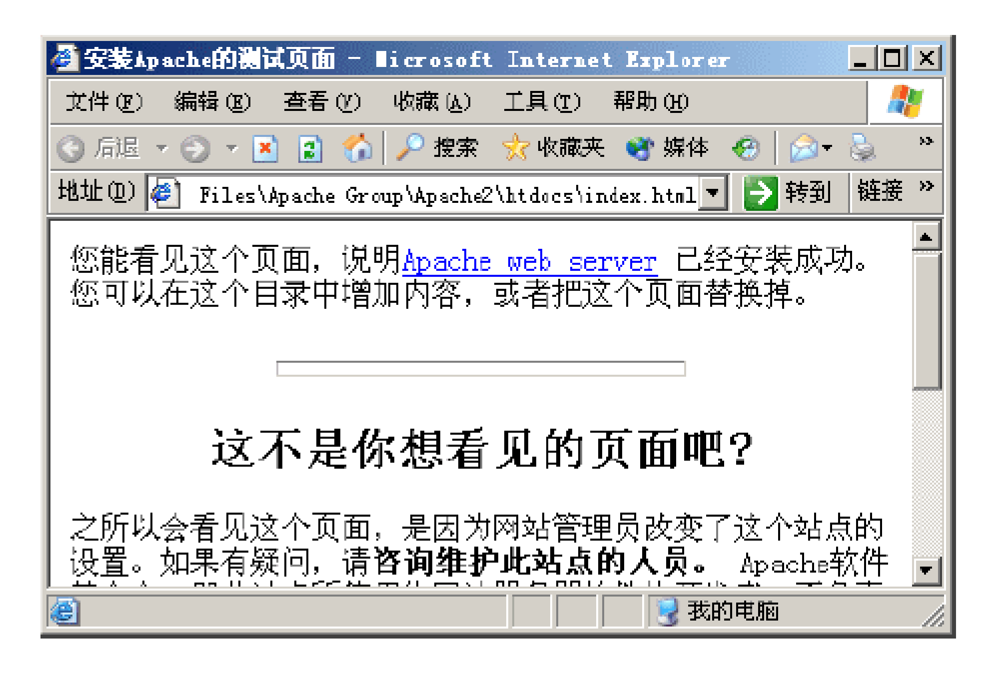


图 4.8　Apache 安装成功信息

需要说明的是，如果计算机上已安装了 IIS，输入“http://localhost”后将首先显示 IIS 中 Web 服务的信息。此时应停止 IIS 中的 Web 服务器，然后再输入“http://localhost”，即可显示 Apache 服务。或者给 IIS 中的 Web 服务指定一个不同于默认的端口号，如 8080；而默认的端口号为 80。此时还可通过“http://localhost：端口号”来继续打开 IIS 中的 Web 服务。

2．PHP 的安装

在安装 PHP 之前需要关闭处于打开状态的 Apache。安装 PHP 有两种方法，一种方法是使用 Windows 安装程序，安装方法同前一节所述，只是在选择服务器类型时应选择 Apache 这一点有所不同。另外一种方法是手工方式安装，安装过程如下：

（1）用 WinZIP 等解压缩工具把 php-5.1.4-Win32.zip 解压到 PHP 安装目录，如 C:\PHP。

（2）把 C:\PHP 目录下的 Projects\php5\Release_TS_inline\php5ts.dll 文件复制到 Windows 系统目录下。

（3）配置 PHP 和 Apache，使之能解析 php 程序。

PHP 的配置方法是：将 C:\PHP\目录中的“php.ini-dist”改名为“php.ini”，并复制到 Windows 系统目录下（如 Windows 2000 系统目录为 C:\winnt\system32）。

Apache 配置方法是：Windows 下有两种方法使 PHP 工作于 Apache 2.0.x 之中。一种是使 CGI 可执行程序，另一种是采用 Apache 模块的 DLL。不管哪种都需要编辑 httpd.conf 来配置 Apache 支持 PHP 并重新启动服务器。

注意：*记住在 Windows 下给 Apache 的配置文件中加入路径值的时候，所有的反斜线例如 c:\directory\file.ext 必须转换成正斜线，如 c:/directory/file.ext。*

如果以 CGI 方式安装 PHP，需要将以下三行加入到 Apache 的 httpd.conf 配置文件中，以设定 CGI：

```
ScriptAlias /php/ "c:/php/"
AddType application/x-httpd-php .php
Action application/x-httpd-php "/php/php-cgi.exe"
```

如果以 Apache 模块方式安装 PHP，需要将以下两行加入到 Apache 的 httpd.conf 配置

文件中，以设定 Apache 2.0 的 PHP 模块：

```
LoadModule php5_module "c:/php/php5apache2.dll"
AddType application/x-httpd-php .php
```

修改 PHP 的配置文件 php.ini，设置 PHP 安装路径参数：

```
PHPIniDir "C:/php"
```

用 PHP 实际所在的路径替换上面命令中的 C:/php/。

（4）重新启动 Apache 服务器，用编辑器编写如下语句：

```
<?
phpinfo();
?>
```

保存文件名为“test.php”到 C:\apache2\htdocs 目录，然后打开浏览器浏览 http://localhost/test.php，出现 PHP 基本信息就说明配置成功（参见图 4.1）。

4.2.3 Windows 下 MySQL 的安装与运行

Windows 下的数据库管理系统的安装相对比较简单，但是必须注意，MySQL 的原始开发平台具有 UNIX 类操作系统的特色，其大部分软件模块具有 UNIX 系统命令风格，只有少数模块才具有 Windows 操作系统的 GUI 界面。安装 Windows 版的 MySQL 基本过程如下：

（1）MySQL for Windows 的文件在 mysql-shareware-4.1.19-win.zip 里面，首先把它解压。

（2）在解压缩后的文件夹中双击 setup.exe。

（3）进入安装程序界面后选择 NEXT，安装程序默认的安装路径是 C:\Program Files\MySQL，可以改变这个设定，然后一直选择 NEXT，并按要求填写内容就可以了。

安装完成后，如启动 MySQL，到开始菜单中找到启动 MySQL 的项即可启动，不需要到安装目录中去执行。

4.3 构建 Linux 的动态 Web 服务器

与 Windows 一样，在 Linux 下也可以架设自己的 Web 服务器，将自己的系统作为服务器，用于发布主页和其他一些信息。目前最流行的 Web 服务器软件是 Apache。实际上，Linux+Apache+PHP 应是最经济的选择，因为这样的组合几乎是不用钱的，成本与效益比也是最好的。许多成功的经验就是采用这种最好的组合。下面详细介绍在 Linux 下如何安装 Apache、PHP 和 MySQL。

4.3.1 安装 Apache

若用户的 Linux 中没有安装 Apache，则需要进行安装。安装分下面两种情况。

1．安装可装载模块和标准的 MPM prefork 版本的 Apache

到网站 http://www.apache.org 下载最新的 Apache 版本，下面以 Apache 2.0.59 版本为例。

在该网站的下载目录中可下载文件 httpd-2.0.59.tar.gz 或 httpd-2.0.59.tar.bz2。httpd-2.0.59.tar.gz 是尚未编译的源代码标准版，需要下载后通过编译生成可执行文件。httpd-2.0.59.tar.bz2 是编译成 Linux 二进制可执行文件版本，只需解压缩即可完成安装。下面讲解如何在 Linux 下安装 Apache。这里省略所有的版本号，以保证本文的正确性。具体安装时需要将本文的“NN”替换为相应的版本号。

1）解压缩

```
gzip -d httpd-2_0_NN.tar.gz
tar -xvf httpd-2_0_NN.tar
gunzip php-NN.tar.gz
tar -xvf php-NN.tar
```

2）转换到解压缩后的目录下

```
cd  httpd-2_0_NN
```

3）配置 Apache

```
./configure --enable-so
```

4）编译和安装 Apache

```
make
make install
```

现在已经将 Apache 2.0.NN 安装在/usr/local/apache2。本安装支持可装载模块和标准的 MPM prefork 版本。之后，可以使用如下命令启动 Apache 服务器。

```
/usr/local/apache2/bin/apachectl  start
```

通过在客户端 Web 浏览器输入 Linux 服务器的 IP 地址进行访问或在本机浏览器地址栏输入 http://127.0.0.1。如果出现 Apache 默认的欢迎网页（参见前面的图 4.8），则表示 Web 服务安装正确并且运行正常。如果成功，可以停止 Apache 服务器并继续安装 PHP。

```
/usr/local/apache2/bin/apachectl  stop
```

2．安装 rmp 版的 Apache

目前几乎所有的 Linux 发行版都捆绑了 Apache，Red Hat Linux 也不例外，在默认情况下 Red Hat Linux 安装程序会将 Apache 安装在系统上。由于目前 Apache 被重新命名为 httpd，因此可使用下面命令检查系统是否安装了 Apache 或查看已经安装的版本。

```
#rpm -q  httpd          //"#"是 Linux 管理员 root 的命令提示符
```

命令执行结果如下所示：

```
[root@RHEL4 ~]# rpm -q httpd
```

```
Httpd-2.0.59-12.ent
[root@RHEL ~]#
```

则表示 Apache 已安装，它的版本为 Httpd-2.0.59-12.ent。

如果系统还未安装 Apache，应将 Red Hat Linux 光盘放入光驱中，并依此执行下列命令。

```
#mount  /dev/cdrom  /mnt/cdrom      //将光驱内容装入/mnt/cdrom 目录
mout: block device /dev/cdrom is write-protected,mout read-only
#cd /mnt/cdrom/RedHat/RPMS/         //进入 rmp 文件所在的目录
#rpm -ivh apache *.rpm              //安装 Apache
```

如要从旧版本升级到新版本，请把参数–i 改为–U 即可。

```
# rpm -Uvh apache *.rpm            //更新 Apache
```

注意：Linux 系统对命令和参数的大小写是敏感的，–A 和–a 的选项意义不同，Linux 命令一般都是小写。

执行/etc/init.d/httpd start 命令启动 Apache 服务器。通过在客户端 Web 浏览器输入 Linux 服务器的 IP 地址进行访问或在本机浏览器地址栏输入 http://127.0.0.1。如果出现 Apache 默认的欢迎网页（见图 4.8），则表示 Web 服务安装正确并且运行正常。

4.3.2 安装 PHP 5

1．安装 rmp 版的 PHP 5

在默认情况下，Red Hat Linux 安装程序会将 PHP 解析器安装在系统上，读者可以使用下面的命令检查系统是否已经安装了 PHP 解析器或查看已经安装了何种版本。

```
#rpm -q  php
```

命令的结果如图 4.9 所示，这表明 PHP 解析器已安装，它的版本是 5.1.4。

```
[root@RHEL4 ~]# rpm  -q  php
Php -5.1.4
[root@RHEL ~]#
```

图 4.9 检查系统是否已经安装了 PHP 解析器

如果系统还没有安装 PHP 解析器，应将安装盘放入光驱，加载光驱后在光盘的 RedHat/RPMS 目录下找到 PHP 解析器的 RPM 安装包文件 php-5.1.4-17.i386.rpm，使用下面的命令安装 PHP 解析器。

```
#rpm -ivh /mnt/cdrom/RedHat/RPMS/php-5.1.4-17.i386.rpm
```

2．安装可装载模块的 PHP

PHP 能够使用大量不同的方法编译，但是最常用的方式是作为 Apache 的可装载模块。Apache 的安装过程在 4.3.1 节已说明，在前面的基础上，下面继续讲解如何配置 PHP。

（1）进入 PHP 源代码目录。

```
cd ../php5-NN
```

（2）用各种各样的参数来配置 PHP。下面给出一个在有 MySQL 支持的 Apache 2 上进行配置的范例。本地的 apxs 的路径可能会不同，假定系统 apxs 的路径被命名为 apxs2。

```
./configure --with-apxs2=/usr/local/apache2/bin/apxs
```

（3）编译和安装 PHP 5。

```
make
make install
```

如果决定在安装后改变配置选项，只需重复（2）～（3）步，然后需要重新启动 Apache 使新模块生效，无须重新编译。

提示：除非明确有提示，否则 “make install” 命令将安装 PEAR、各种 PHP 工具诸如 phpize 等。

（4）配置 php.ini。

```
cp php.ini-dist /usr/local/lib/php.ini
```

编辑 php.ini 文件以修改 PHP 的选项。如果想要把此文件放到另外的位置，需要在步骤（2）添加--with-config-file-path=/path 选项。

如果选择 php.ini -recommended，请务必阅读其中的变更列表，它们将影响 PHP 的执行。

（5）编辑 Apache 的 httpd.conf 文件以调用 PHP 模块。命令如下：

```
LoadModule php5_module modules/libphp5.so
```

LoadMolude 表达式右边的路径必须指向系统中的 PHP。以上的 make install 命令可能已经帮用户完成了这些，但务必要检查。

（6）告知 Apache 将特定的扩展名解析成 PHP 程序，例如，让 Apache 将扩展名 php 解析成 PHP 程序文件。可以将任何扩展名指定为 PHP，只需添加它们，每一个用空格分隔。例如，添加.php 和.phtml 扩展名。

```
AddType application/x-httpd-php .php .phtml
```

（7）启动 Apache 服务器。

```
/usr/local/apache2/bin/apachectl start
```

按照上面的步骤，便可以使 Apache 2.0 将 PHP 作为 Apache 可转载的 SAPI 模块。当然 Apache 和 PHP 都还有很多配置选项，可以在相应的源代码目录中使用./configure-help 获得更多信息。

4.3.3 Linux 下 MySQL 的安装与运行

几乎所有的 Linux 发行版本都内置了 MySQL 数据库，Red Hat Enterprise Linux 也不例外，它内置了 MySQL4.1.10a，只不过系统安装程序默认安装了 MySQL 的客户程序。可使

用下面的命令检查系统是否已经安装了 MySQL 或查看已经安装了何种版本。

```
#rmp -qa | qrep mysql
```

命令的执行结果如图 4.10 所示。

```
[root@RHEL4 ~]# rpm  -qa  | qrep mysql
 mysqlclient10-3.23.58-4.RHEL4.1
 mysqlclient10-3.23.58-4.RHEL4.1.i386.rpm。
[root@RHEL ~]#
```

图 4.10　检查系统是否安装了 MySQL

从图 4.10 中可见，系统当前仅安装了 MySQL 的客户程序，其中：

mysql 是客户程序及共享库，其对应的 PRM 包文件为 mysqlclient10-3.23.58-4.RHEL4.1。

mysqlclient10 是用来兼容 3.23.x 版本客户端库，其对应的 RPM 包文件为 mysqlclient10-3.23.58-4.RHEL4.1.i386.rpm。

如果还没有安装 MySQL，如要使用数据库服务，通常还需要在 Red Hat Enterprise Linux 的安装盘中找到并安装以下 RPM 包文件：

- mysql-server-4.1.10a-2.RHEL4.1.i386.rpm（MySQL 数据库服务）；
- mysql-bench-4.1.10a-2.RHEL4.1.i386.rpm（MySQL 数据库基准和性能测试工具）；
- mysql-devel-4.1.10a-2.RHEL4.1.i386.rpm（开发 MySQL 程序时使用的库和头文件）；
- mysqlclient10-3.23.58-4.RHEL4.1.i386.rpm（开发 MySQL 客户端程序时使用的库和头文件）。

要安装上述 RPM 包文件，可将安装盘放入光驱，加载光驱后在光盘的 RedHat/RPMS 目录下找到相应的 RPM 安装包文件，然后使用 rpm –ivh 命令安装。例如，要安装 MySQL 数据库服务，可使用下面的命令：

```
#rpm -ivh /mnt/cdrom/RedHat/RPMS/mysql-server-4.1.10a-2.RHEL4.1.i386.rpm
```

命令执行后，如果出现如图 4.11 所示的结果，则表示 mysql-server 安装成功。

```
[root@RHEL4 ~]#rpm -ivh /media/cdrom/RedHat/RPMS/mysql
    -server-4.1.10a-2.RHEL4.1.i386.rpm
Warning:/media/cdrom/RedHat/RPMS/mysql-server-4.1.10a
-2.RHEL4.1.i386.rpm:V3 DSA signature;NOKEY,key ID db42a60e
Preparing…     ##################################################[100%]
1:mysql-server ##################################################[100%]
[root@RHEL 4~]#
```

图 4.11　安装 MySQL 服务

如果使用的安装包是 tar.z，就先用 uncompress 解压。然后在/user 或者/home 目录下用

tar-xvf filename 命令安装。

如果使用的安装包是.rpm 并且操作系统是 RedHat，可以在 X-Windows 下的文件管理器中直接双击这个文件进行安装。

如果使用的安装包是.tar.gz，那就要先用 gunzip 解压，然后安装：

```
gunzip  mysql-xxxx.tar.gz|tar -xvf
```

安装完成后，重新启动操作系统，MySQL 就会自动启动了。

实　验　4

1．给一台计算机安装 Linux 系统。
2．基于一种 Windows 操作系统建立 Web 服务器支持 PHP 运行环境。
3．基于一种 Linux 操作系统建立 Web 服务器支持 PHP 运行环境。
4．在第 2 题或第 3 题的基础上搭建一台 MySQL 数据库服务器。

习　题　4

1．简述 PHP 的发展历史。
2．PHP 5 有哪些新增的特性？
3．如何才能获取本地主机（localhost）的 IP 地址？
4．127.0.0.1 是一个什么样的 IP 地址，有何用途？
5．在 Windows 和 Linux 下有几种搭建 PHP 运行环境的方法？
6．请叙述 Apache、PHP、MySQL 的运行原理及三者之间的关系。
7．要使 Apache、PHP、MySQL 能够正常执行和运行，一般需修改哪两个文件内容？
8．如何设置才能让 Apache 服务器能够处理其他扩展名如.php5、phtml 的 PHP 程序？

第 5 章　PHP 5 的程序设计基础

本章导读

PHP 5 是目前最流行的网页脚本语言之一，是属于服务器端 HTML 嵌入式的脚本语言。它在保证最大可操作性的前提下，提供了比一般 CGI 更快的执行速度，多平台特性使其可以无缝地运行在 UNIX 和 Windows 平台，更为突出的功能是它对数据库的操作能力，强大的兼容性使 PHP 可以操作几乎所有的数据库，并且在对数据库操作的简便性上得到了绝大多数人的认可。本章结合 PHP 的基本概念、基本语法等各个方面的实际应用，将 PHP 5 的内容全面展开。

5.1　PHP 5 程序的语法结构

5.1.1　一个简单的 PHP 5 程序示例

为了对 PHP 5 有一个基本的认识，这里先从第一个最简单的，用 PHP 5 编写的实例程序“hello,world”开始体会嵌入的含义。

【例 5.1】 显示“hello,world”的程序代码。

```
<HTML>
<HEAD>
<TITLE> Hello World!</TITLE>
</HEAD>
<BODY>
<H1> <CENTER>
<?php
$string="hello,world!";
echo $string;
?>
</CENTER></H1>
</BODY>
</HTML>
```



图 5.1　最简单的 PHP 程序

访问该程序，在浏览器上将显示如图 5.1 所示的结果。

由 PHP 代码生成的输出将替代<?php…?>标记。通过浏览器查看其源文件，将看到如下所示的内容。

```
<HTML>
<HEAD>
<TITLE> Hello World! </TITLE>
</HEAD>
<BODY>
<H1> <CENTER>
Hello,World!
</CENTER></H1>
</BODY>
</HTML>
```

通过这个程序，应该认识到 PHP 5 语言是在 Web 服务器端运行的。当开发互联网应用程序时，很重要的一点是要清楚应用程序是在哪一端运行的。PHP 5 总是运行在服务器端。由于 Java Applet、ActiveX 或 JavaScript 脚本都是运行在客户端的 Web 浏览器中的，因此 PHP 5 不能直接和它们进行比较。然而这些语言可以很容易地与 PHP 结合在一起，使用 PHP 可以很方便地生成所需要的任何 HTML 代码，当然也能激活 Java applet 和 ActiveX 控件，此外，它还可以动态生成 JavaScript 语句。

5.1.2　PHP 5 程序嵌入网页的方法

从语法上看，PHP 语言是借鉴 C 语言的语法特征，由 C 语言改进而来的。人们可以混合编写 PHP 5 代码和 HTML 代码，不仅可以将 PHP 5 脚本嵌入到 HTML 文件中，还可以把 HTML 标签也嵌入在 PHP 5 脚本里。

1．HTML 文档嵌入 PHP 代码的方法

在 HTML 文档中嵌入 PHP 代码的常用方法有三种:

（1）<? echo("PHP 程序代码");?>

（2）<?php echo("PHP 程序代码");?>

（3）<script language="php">
　　echo ("PHP 5 程序代码");
　　</script>

其中第一种及第二种是最常用的两个方法，在小于号加上问号后，可以加也可以不加 php 三个字符，之后就是 PHP 的代码。在代码结束后，加入问号和大于号就可以了。第三种方法对熟悉 Netscape 服务器产品的网页制作人员而言，有相当的亲切感，它是类似 JavaScript 的写法。

其实，编混合程序最好的方法，就是先写纯 HTML 格式的文件，再将需要用变量或进行其他处理的地方改成程序。这种方法，可以让用户在开发时达到事半功倍的效果。

2．语句分隔符

PHP 与 C 或 perl 的程序语法约定相同，语句之间用分号分隔。但是 PHP 标记的结束符（?>）同时也意味着最后一个语句的结束，所以，以下两个程序片段是等价的。

程序 1：

```
<?php
echo "php5 程序代码";
```

```
?>
```

程序 2：

```
<?php echo "php5 程序代码" ?>
```

3．注释

在 PHP 程序中，加入注释的方法很灵活。可以使用 C 语言、C++语言或者 UNIX 的 Shell 语言的注解方式，而且也可以混合使用。PHP 支持 C、C++和 UNIX 风格的注释方式：

```
/* 这是 C、C++风格多行注释 */
// 这是 C++风格的单行注释
# 这是 UNIX Shell 风格的单行注释
```

5.2 PHP 5 的数据类型

本节讲述 PHP 5 语言中的数据类型。每一种数据类型描述了该语言支持的一组对象和对对象进行的操作。PHP 5 支持整数、浮点数、布尔数、字符串、数组和对象。变量类型通常不由程序员决定而由 PHP 5 运行过程决定。

数据类型是语言定义的数学抽象，PHP 5 支持八种原始类型。

1．四种标量类型

- 布尔型（Boolean）：这是最简单的类型。Boolean 表示真或假值，可以为 TRUE 或 FALSE。
- 整型（integer）：在 32 位的操作系统中，它的有效范围是–2 147 483 648～+2 147 483 647。要使用十六进制整数可以在前面加 0x。
- 浮点型（float，也作“double”）：在 32 位的操作系统中，它的有效范围是 1.7E–308～1.7E+308。
- 字符串（string）：无论是单一字符或数千字的字符串都是使用这个变量类型。值得注意的是，要指定字符串给字符串变量，需要在头尾加上双引号。

2．复合类型

- 数组（array）：可以是二维、三维或者多维数组，数组中元素的类型可以是 string、integer 或是 double 类型。
- 对象（object）：为类，目前在 PHP 5 中的类不多。

3．两种特殊类型

- 资源（resource）：资源是一种特殊变量，保存了到外部资源的一个引用。
- NULL：特殊的 NULL 值表示一个变量没有值。

5.2.1 数值

数值可以是整型或浮点型。可以用以下的语句形式将一个数值赋值给变量。

```
$a = 1234;        # 十进制数
$a = 0123;        # 八进制数 （等于十进制数的 83）
$a = 0x12;        # 十六进制数(等于十进制数的 18)
```

```
$a = 1.234;      # 浮点数
$a = 1.2e3;      # 双精度数的指数形式
```

5.2.2 字符串

字符串可以是由单引号或双引号括住的一串字符。指定一个简单字符串的最简单的方法是用单引号（'）括起来。如要表示一个单引号，需要用反斜线（\）转义。如果在单引号之前或字符串结尾需要出现一个反斜线，需要用两个反斜线表示。

如果用双引号（"）括起字符串，PHP 能够识别更多特殊字符的转义序列。常用的转义字符如表 5.1 所示。

表 5.1 转义字符

转义字符	含义
\n	换行符（LF 或 ASCII 值 0x0A（10））
\r	回车符（CR 或 ASCII 值 0x0D（13））
\t	水平制表符（ASCII 值 0x09（9））
\\	反斜线（\）
\$	美元符号
\"	双引号
\[0-7]{1，3}	此正则表达式序列匹配一个用八进制符号表示的字符
\x[0-9A-Fa-f]{1，2}	此正则表达式序列匹配一个用十六进制符号表示的字符

此外，如果试图转义任何其他字符，反斜线本身也会被显示出来。用双引号括住的字符串，最重要的一点是其中的变量名会被变量值替代。

注意：单引号与双引号使用上不同在于，被单引号括住的字符串是以字面定义的，而双引号括住的字符串可以被扩展。以下是对字符串赋值的例子：

```
$first = 'Hello';
$second ="World";
$full1 = "$first $second"; # 产生 Hello World
$full2 = '$first $second'; # 产生 $first $second
```

5.2.3 布尔型

这是最简单的类型。boolean 类型表示真或假，其值可以为 TRUE 或 FALSE。要指定一个布尔值，使用关键字 TRUE 或 FALSE（两个都是对大小写不敏感）。

```
<?php
$a1 = True;   // 将值 TRUE 赋予变量 a1
?>
```

5.3 PHP 5 的常量和变量

5.3.1 常量

常量是一个简单值的标识符（名字）。在脚本执行期间该值不能改变。常量名默认为

区分大小写。按照惯例常量标识符总是大写的。

常量名和其他任何 PHP 标识符遵循同样的命名规则。合法的常量名以字母或下划线开始，后面跟着任何字母、数字或下划线。

PHP 本身定义了以下一些内置常量。

- __FILE__：该常量是当前执行的 PHP 程序文件名。若执行到引用文件（include 或 require），则该常量为引用文件名，而不是引用它的文件名。
- __LINE__：该常量是执行的 PHP 程序行数。若执行到引用文件（include 或 require），则该常量为引用文件的行数，而不是引用它的文件的行数。
- PHP_VERSION：该内置常量是 PHP 程序的版本，如‘5.1.4’。
- PHP_OS：该常量指执行 PHP 解析器的操作系统名称，如‘Linux’。
- TRUE：该常量就是真值（true）。
- FALSE：该常量就是假值（false）。
- E_ERROR：该常量指到最近的错误处。
- E_WARNING：该常量指到最近的警告处。

当然在程序写作时，以上的 PHP 内置常量是不够用的。在 PHP 中，允许用户用 define() 函数自行定义所需要的常量。如下例：

```
<?php
define("IP","192.168.1.2");
echo IP;
?>
```

注意：一个常量一旦被定义，就不能再改变或者取消定义。

在程序中，可以通过指定常量名字来取得常量的值，不要在常量名前面加上$符号。

5.3.2 变量

PHP 中一个美元符号（$）后面跟上一个变量名称，即表示一个变量。变量名与 PHP 中其他标识符一样遵循相同的规则。一个有效的变量名由字母或者下划线开头，后面跟上任意数量的字母、数字或者下划线。需要注意的是，变量的名称是区分大小写的。

当需要在程序执行期间存储一些运算结果时，就需要使用变量。PHP 有三种类型的变量（标量、数组和对象）。数组在 5.7 节介绍，对象在第 6 章介绍。

标量是用来存储一个信息的变量。例如，$int_x、$str_name 都是合法的标量。

在 PHP 5 中初始化一个标量，只需要简单地分配给它一个值即可。例如：

```
$int_number=46;
$str_title="I love php5";
```

5.4 PHP 5 的运算符和表达式

运算符可以用来处理数字、字符串及其他需要比较运算的条件。PHP 5 具有 C、C++ 和 Java 中的通常见到的运算符。这些运算符的优先权也是一致的。赋值同样使用“=”。PHP 5

的运算优先顺序可以参见表 5.2，在混合式的情形下，愈往下表示优先级愈高。

表 5.2　运算符的结合规则表

运算符	结合规则	运算符	结合规则
,	左至右	^	左至右
or	左至右	&	左至右
xor	左至右	==,!=	不限
and	左至右	<,<=,>=,>	不限
.=,&=,|=,/=,%=,^=,=,+=,–=,*=	左至右	<<,>>	左至右
?,:	左至右	+,–,	左至右
||	左至右	*,/,%	左至右
&&	左至右	!,~,++,– –,@	右至左
|	左至右	[]	右至左

1．算术运算符

算术运算符就是用来处理四则运算的符号，表 5.3 列出了常用的算术运算符及其意义。

表 5.3　算术运算符

符号	意义	符号	意义
+	加法运算	–	减法运算
*	乘法运算	/	除法运算
%	取余数	++	累加
– –	递减		

【例 5.2】　算术运算符的应用。

```
<?php
$a = 8;
$b = 2;
$c = 3;
echo $a+$b."<br>\n";
echo $a-$b."<br>\n";
echo $a*$b."<br>\n";
echo $a/$b."<br>\n";
echo $a%$c."<br>\n";
$a++;
echo $a."<br>\n";
$c--;
echo $c;
?>
```

图 5.2　例 5.2 运行结果的界面

该程序运行的结果如图 5.2 所示。

2．字符串运算符

字符串运算符号只有一个，就是英文的句号“.”。它可以将字符串连接起来，变成一

个合并后的新字符串。

【例 5.3】 字符串运算符的应用。

```
<?php
$a = "PHP 5";
$b = "功能强大";
echo $a.": ".$b;   //输出结果为 PHP 5: 功能强大
?>
```

3．比较运算符

比较运算符用来比较两个相同类型的数据的大小、是否相等的关系，比较的结果是一个布尔型，值为 TRUE 或 FALSE。表 5.4 列出了比较运算符及其意义。

表 5.4 比较运算符

符号	意义	符号	意义
<	小于	>=	大于或等于
>	大于	==	等于
<=	小于或等于	!=	不等于

【例 5.4】 比较运算符的应用。

```
<?
$a= 5;
if($a!=5) {
  echo "a 不是 5";
} else {
  echo "a 是 5";
}
?>
```

4．逻辑运算符

逻辑运算符用来连接一个或两个布尔型的数据或表达式，逻辑运算的结果是真或假。通常，逻辑运算符用来表示一个复杂的条件。表 5.5 列出了逻辑运算符及其意义。

表 5.5 列出了逻辑运算符

符号	意义	符号	意义
&& 或 and	与（And）	|| 或 or	或（Or）
xor	异或（Xor）	!	非（Not）

在 PHP 中，数值 0 表示逻辑假（FALSE），非 0 的任何一个数都表示逻辑真（TRUE）。

5．赋值运算符

赋值运算符用来将一个表达式的值赋给一个变量，表 5.6 列出了赋值运算符及意义。

下面是一个赋值运算符的应用例子。

表 5.6 赋值运算符

符号	意义	符号	意义
=	将右边的值连到左边	+=	将右边的值加到左边
−=	将右边的值减到左边	*=	将左边的值乘以右边
/=	将左边的值除以右边	%=	将左边的值对右边取余数
.=	将右边的字符串加到左边		

```
<?
$a=8;
$a+=7;   //即$a=$a+7;
echo $a;   //输出 15
?>
```

6．其他运算符

除了上述的运算符号外，还有一些运算符号难以归类，表 5.7 列出了这些特殊的运算符。

表 5.7 其他运算符

符号	意义	符号	意义
$	变量	&	变量的内存地址（加在变量前）
@	不显示错误信息（加在函数前）	->	类的方法或者属性
=>	数组的元素	?.:	三元运算符

其中，比较特殊的是三元运算符，以下例来解释。

```
(expr1) ? (expr2) : (expr3);
```

它表示：若 expr1 的运算结果为 true，则执行 expr2；否则执行 expr3。实际上它有点类似 if…else 语句，但可以让程序较精简有效率。

7．表达式

当操作数和操作符组合到一起时，它们即组成一个表达式。表达式是由一个操作符或多个操作符将操作数连接起来的式子。最基本的表达式就是数字，如 12。下面是一些表达式的例子：

```
-12+14
-12+14*(24/12)
(-12+14*(24/12)) && $myname == "Chen"
(10+2)/(count_star()*114)
```

提示：书写表达式时，有时很难分清左括号和右括号的数目是否相同。从左到右，当左括号出现时，就加一，当右括号出现时，就从总数中减一。如果在表达式的结尾时，总数为零时，左括号和右括号的数目就一定相同。

5.5 PHP 5 程序的数据输入和输出

5.5.1 数据输出

PHP 的大多数程序是安静地执行的，完成某一计算功能，它们不显示任何输出。嵌入的 PHP 程序通过输出语句向 HTML 网页输出结果，作为网页的一部分内容，再发送到用户浏览器，从而在用户浏览器中看到 PHP 程序输出的结果。可以使用 echo 或 print 函数输出信息。

打印输出的两种最基本函数是 echo 和 print。在实际使用中，print 和 echo 两者的功能几乎是完全一样。但是，两者之间也存在一个非常重要的区别：在 echo 函数中，可以同时输出多个字符串，而在 print 函数中则只可以同时输出一个字符串。同时，echo 函数并不需要圆括号，所以 echo 函数更像是语句而不像是函数。

1．回显

echo 的最简单用法是打印作为参数的字符串，例如：

```
echo "this will print in the user's browser windows";
```

它与下面语句是等价的：

```
echo("this will print in the user's browser windows");
```

这两条语句的结果都是输出同一行英文：

```
this will print in the user's browser windows
```

2．打印

print 函数非常类似于 echo，但有两点重要区别：一是 print()可接受一个参数。二是 print()返回一个值，表示 print 语句执行是否成功。如果执行打印成功，print()返回的值为 1，如果不成功则返回 0。

例如，下面的两行将打印同样的结果：

```
print("3.14159");
print(3.14159);
```

5.5.2 数据输入

不管是什么程序语言，数据输入是不可缺少的。C、VB 等其他编程语言本身提供了输入函数或者文本框等控件，完成程序运行过程中的数据输入。但是对于 PHP 脚本语言来说，并没有提供数据输入的语句或函数。因为 PHP 是 Web 编程语言，PHP 程序的所有数据输入都来自于用户的浏览器。因此，PHP 的数据输入可以通过以下两种方法来实现。

第一种方法是通过在浏览器地址中输入网址后面附带输入数据的方式来实现，例如：

```
http://localhost/echoexample.php?name=Chen&love=tennis
```

在所访问的文件名之后，以问号（？）开始，其后面的内容给出了需要传递给 PHP 程

序的数据。在此例中，位于“=”号左边的称为“参数”，位于等号右边的则是“参数值”。参数是 PHP 程序读取数据的根据，参数名要根据 PHP 程序中的变量名而定，参数值则可依需要加以改变。下面给出 echoexample.php 的程序代码。

【例 5.5】 echoexample.php 的程序代码。

```
<HTML> <HEAD>
<TITLE> How are you!</TITLE>
</HEAD>
<BODY>
<?php
  $string1=$_GET["name"];
  $string2=$_GET["love"];
  echo $string1;
  echo ",您喜欢的体育运动是: ";
  echo $string2;
?>
</BODY> </HTML>
```

该程序使用了 PHP 的全局数组$_GET。$_GET 数组存储来自于用户浏览器地址栏中所有参数的值，在此例中，$_GET["name"]元素存储 name 参数的值，$_GET["love"]元素存储 love 参数的值。图 5.3 是该程序的一次访问结果。



图 5.3 访问 echoexample.php 的输出

注意：浏览器地址栏中输入各个参数名必须与 PHP 程序中相应的$_GET 数组的元素下标名相同，而且字母的大小写也要一致。

如果将浏览器地址的网址中的 name、love 参数值做改变，如下：

```
http://localhost/CH05/echoexample.php?name=Feng&love=basketball
```

则结果会随之发生改变。这是因为所输入的参数值不同，所以结果也就不同。

第二种方法是利用表单来输入数据，表单输入方式比利用网址来输入数据更具亲和力。下面通过几个例子来说明表单输入的 PHP 编程。

首先以一个简单的示例来说明表单和 PHP 程序的使用。图 5.4 是一个调查页面，用户输入信息后，单击页面上的“确定”按钮，将数据发送到 Web 服务器的脚本，由脚本处理收到的数据，生成一个新网页，返回给用户的浏览器，结果如图 5.5 所示。

图 5.4 和图 5.5 所显示的过程是通过下面例 5.6 和例 5.7 的代码来实现的。

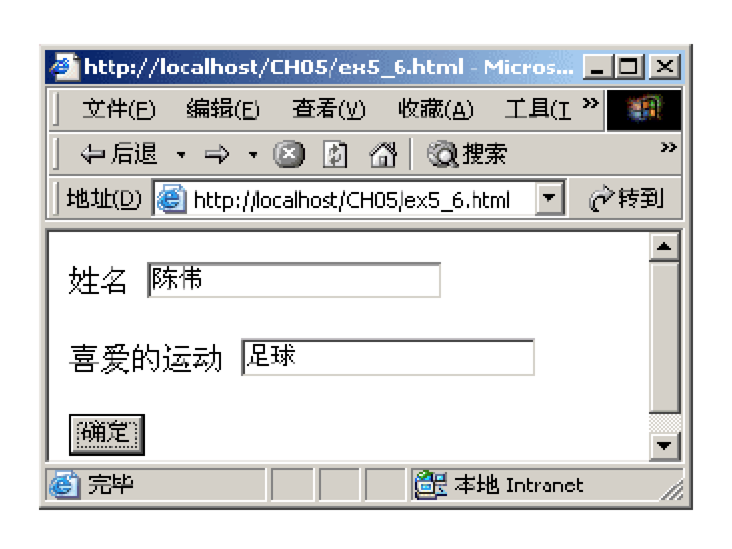




图 5.4 调查页面



图 5.5 返回的确认页面

【例 5.6】 简单的 HTML 表单，文件名为 ex5_6.html。源代码如下：

```
<html><head></head>
<body>
<form name="form1" method="post" action="ex5_7.php">
  <p>姓名 <input name="name" type="text" id="name"> </p>
  <p> 喜爱的运动 <input name="love" type="text" id="love"> </p>
  <p> <input type="submit" name="Submit" value="确定"> </p>
</form>
</body></html>
```

下面对表单标记<form>的属性进一步说明。method 属性指定发送表单数据的方式，一般选择为 post 或 get。两者的区别是：提交表单时，使用 get 方式，表单上输入的信息作为字符串附加在 action 所设定的 URL 之后，中间用“？”和“&”隔开，然后发送到 Web 服务器。使用 post 方式时，表单上输入的信息则不是附加在 action 所设定的 URL 之后发送。action 属性指定接收表单数据的服务器端程序，本例指定为 ex5_7.php，即下面的例 5.7 的程序。

【例 5.7】 接收表单数据的 PHP 程序，文件名为 ex5_7.php。程序如下：

```
<html><head></head>
<body>
<?
echo $_POST["name"];
echo ",您喜爱的运动是: ".$_POST["love"];
?>
</body>
</html>
```

在该程序中，使用 PHP 内置的全局数组变量$_POST。$_POST 数组存储了用户浏览器中表单上的各个表单元素的值，其元素的下标名必须与网页中表单的元素名称相同，而且大小写也一致。

在 HTML 表单中，除了使用文本框元素外，还可以使用单选按钮、复选框等表单元素来输入数据。对于同一组单选按钮，它们的名字必须相同。而对于同一组的复选框，它们的名字也要相同，并在其名字后加上一对方括号（[]）。这样 PHP 脚本程序把这些选中的复选框的值作为一个数组来处理。下面用例 5.8 和例 5.9 说明这两种按钮的使用。

【例 5.8】 含有复选框、单选按钮的 HTML 表单，文件名为 ex5_8.html。源代码如下：

```
<html><head>
<title>调查问卷</title>
</head>
<body>
<form name="form1" method="post" action="ex5_9.php">
  <p>姓名 <input name="name" type="text"> <br>
    性别
    <input name="sex" type="radio" value="男" checked>男
    <input type="radio" name="sex" value="女">女
    <br>感兴趣的编程语言
    <input name="choose[]" type="checkbox" value="PHP">PHP
    <input name="choose[]" type="checkbox" value="VB">VB
    <input name="choose[]" type="checkbox" value="VC++">
    VC++ </p>
  <p> <input type="submit" name="Submit" value="确定"></p>
</form>
</body></html>
```

【例 5.9】 接收并处理复选框、单选按钮的 PHP 程序，文件名为 ex5_9.php。程序如下：

```
<html><head></head>
<body>
<?
echo $_POST["name"];
echo "你的性别是 ".$_POST["sex"];
echo ",感兴趣的编程语言是: <br>";
for ($i=0;$i<count($_POST["choose"]);$i++) {
   echo $_POST["choose"][$i]."  ";
}
?>
</body></html>
```

首先访问 ex5_8.html，显示如图 5.6 所示的页面，输入用户信息后，单击“确定”按钮，将表单数据传送给 Web 服务器端的 ex5_9.php，由它进行处理，生成新网页，返回给用户浏览器，结果如图 5.7 所示。



图 5.6 ex5_8.html 的页面

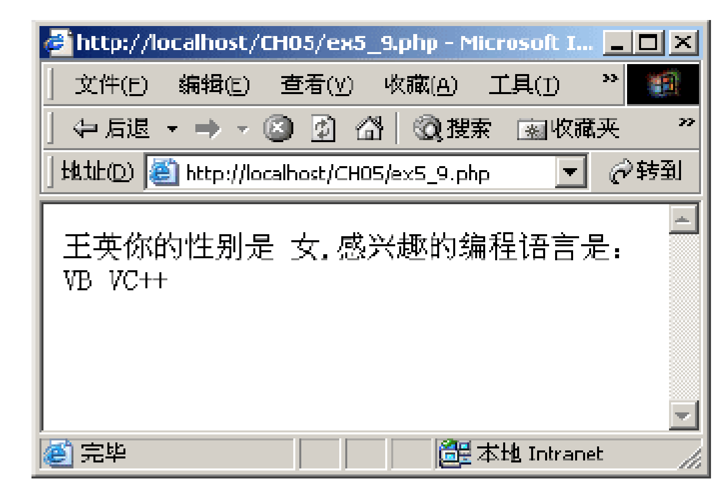


图 5.7 返回的调查结果

5.5.3 赋值语句

基本的赋值运算符是“=”，它将运算符左侧的变量值设为右侧表达式的值。赋值表达式的值就是被赋予的值。这样，就可以编写一些复杂的语句。如：

```
$a=($b=8)+8;      //$a 现在的值是 16,$b 的值是 8
```

除基本运算符之外，前面给大家介绍的赋值运算符也可在赋值语句中使用。下面给出一些赋值语句的示例。

```
<?php
  $a = 3;
  $a += 5;
  $b = "Hello ";
  $b.= "There!";
?>
```

5.6 PHP 5 程序的流程控制语句

任意一个脚本程序都是由一系列语句组成的，语句可以是一个赋值语句、一个函数调用、流程控制语句，甚至还可以是一个什么都不做的空语句，语句通常用分号结尾。此外，可以用将一组语句括起来，形成一个语句组。流程控制语句可以控制程序执行的路径，它有三种基本的流程结构：顺序结构、分支结构和循环结构。下面介绍 PHP 的分支和循环。

5.6.1 分支结构语句

1. if 语句

PHP 5 中 if 语句的格式与 C 语言的相似，但也有个别不同的地方。if 语句有四种形式。最简单的 if 语句形式如下：

```
if (表达式) {
  语句序列
}
```

这种形式的 if 语句只带 if 子句，表达式代表一个判断条件。解释为“如果表达式的值为真，则执行语句序列的命令”。

【例 5.10】 最简单的 if 语句示例。

```
<?php
$a=1;
$b=1;
if ($a==$b) {
  echo "a 等于 b";
}
?>
```

第二种形式的 if 语句是除了带上 if 子句外，还加上 else 子句，可解释成“若发生了某事则怎样处理，否则该如何解决”。其语法如下：

```
if (表达式) {
  语句 1
} else {
  语句 2
}
```

【例 5.11】 利用 if…else 语句完善例 5.10 的程序。

```
<?php
$a=1;
$b=1;
if ($a==$b) {
  echo "a 等于 b";
} else
 echo "a 不等于 b";
?>
```

第三种形式的 if 语句就是带有多个 elseif 子句的 if 语句，通常用在判断条件有多种可能的情形。其语法如下：

```
if (表达式 1) {
    语句 1
}
elseif (表达式 2) {
    语句 2
}
elseif (表达式 3) {
    语句 3
}
⋮
elseif (表达式 n) {
    语句 n
}
else {
    语句 n+1
}
```

这种 if 语句按照表达式的顺序，逐个地判断表达式的值是否真，如果检测到某一个表达式为真，则执行与之相应的语句。如果所有表达式都为假值，则执行 else 子句中的语句。

【例 5.12】 嵌套的 if 语句的应用。

```
<?php
$a=1;
$b=1;
```

```
if ($a > $b) {
  echo "a 比 b 大";
} elseif ($a == $b) {
  echo  "a 等于 b";
} else {
  echo "a 比 b 小";
}
?>
```

第四种形式就是 if 语句的交替格式：if():…..endif。PHP 允许用另外一种办法在 if 语句中使用一组语句。这通常用于嵌套一段 HTML 代码块于 if 语句里，但也可以用于任何地方。if()后面必须用一个冒号取代原来的大括号，后跟一条或几条语句组成的语句组，并以 endif 结束。例如：

```
<?php if($a == 8):?>
A=8
<?php endif:?>
```

2．switch 语句

switch 语句用来处理复杂的条件判断，每个子条件都是 case 指令部分。在实际应用中，若使用许多类似的 if 指令，可以将它们改成 switch 语句。其语法如下：

```
switch (表达式) {
  case 常量 1:
    语句 1;
    break;
  case 常量 2:
    语句 2;
    break;
   ⋮
 default:
    语句 N;
}
```

冒号后为符合该条件要执行的语句组。注意，要用 break 来终止 switch 语句的执行，并控制流程退出 switch 语句。

【例 5.13】 显示今天日期的实际星期数。

```
<?php
switch (date("D")) {
  case "Mon":
    echo "今天星期一";
    break;
  case "Tue":
    echo "今天星期二";
    break;
```

```
    case "Wed":
      echo "今天星期三";
      break;
    case "Thu":
      echo "今天星期四";
      break;
    case "Fri":
      echo "今天星期五";
      break;
    default:
      echo "今天放假";
      break;
}
?>
```

很明显，上述例子用 if 语句就很麻烦了。当然在设计时，要将出现几率最大的条件放在最前面，最少出现的条件放在最后面，可以增加程序的执行效率。上例由于每天出现的几率相同，所以不用注意条件的顺序。

5.6.2 循环结构语句

1．while 循环

while 循环用来在指定的条件内，不断地重复执行指定的语句。其语法如下：

```
while (表达式) {
  语句组
}
```

【例 5.14】 在浏览器里显示 10 行“今天是国庆节”。

```
<?php
$i = 1;
while ($i <= 10) {
  print $i++;
  echo "今天是国庆节<br>\n";
}
?>
```

2．do…while 循环

do…while 循环是先执行，再判断是否要继续执行，也就是说，循环至少被执行一次。这种的循环和 while 循环是不同的（while 循环是先判断再执行）。其语法如下：

```
do {
  语句组
} while (表达式);
```

【例 5.15】 do…while 语句的应用。

```
<?php
$index=0;
Do{
    echo "Inside Do Statement:$index<br>";
    $index++;
}while($index<5);
echo "Outside Do Statement:$index<br>";
?>
```

该程序的运行结果是：

```
Inside Do Statement:0
Inside Do Statement:1
Inside Do Statement:2
Inside Do Statement:3
Inside Do Statement:4
Outside Do Statement:5
```

3．for 循环

for 循环是用得非常多的一种循环，它和 C 语言中相应语句的使用方法相同，其语法如下：

```
for (表达式 1; 表达式 2; 表达式 3) {
  语句组
}
```

表达式 1 将在循环的开始处无条件地计算执行一次。表达式 2 在每次循环的开始处都会被计算。如果表达式 2 的值为 true，则循环将继续，同时继续执行嵌套的语句；如果其值为 false，循环的执行就结束了。表达式 3 是在每次执行完循环体后执行的，常用来改变有关变量的值。

【例 5.16】 用 for 循环写的显示 10 次“今天是国庆节”。

```
<?php
for ($i=1; $i<=10; $i++) {
  echo $i." 今天是国庆节<br>\n";
}
?>
```

从上例可以看到 for 循环和 while 循环的不同。实际应用中，若循环有初始值，且每次循环时都要累加（或累减），则使用 for 循环比用 while 循环好。

5.6.3 跳转语句

跳转语句有两个：break 和 continue。break 语句的功能在于用来跳出目前执行的循环。

【例 5.17】 break 语句的应用。

```
<?php
```

```
$i = 0;
while ($i < 5) {
  if ($arr[$i] == "stop") {
    break;
  }
  $i++;
}
?>
```

continue 语句的功能在于立即停止目前执行循环，并回到循环的条件判断处。

【例 5.18】 continue 语句的应用。

```
<?php
while (list($key,$value) = each($arr)) {
  if ($key%2) { // 略过偶数
    continue;
  }
  do_something_odd($value);
}
?>
```

5.7 PHP 5 的数组

数组就是把一系列数字和字符串作为一个单元来处理。数组中的每一个信息都被认为是数组的一个元素。比如可以用数组存储一个文件中的所有行或者存储一个地址列表。

1．给数组赋初值

对数组元素来说，有三种方法可以设置其初始值，可以对每一个元素分别赋值：

```
$arr_zoo["pelican"] = "Bird with a big beak,";
$arr_zoo["cheetah"]="fast cat.";
$arr_zoo["horse"] = "four-legged animal,";
```

也可以用 array()函数同时对多个元素赋值：

```
$arr_zoo=array(
"pelican"=>"Bird with a big beak,",
"cheetah"=>"fast cat.",
"horse"=>"four-legged animal,");
```

最简单的赋值方法是在数组的下一个空位上增加一个元素，第一个位置是 0，第二个位置是 1，以此类推。例如下面的代码给$arr_names 数组增加了三个元素，这三个元素的下标分别为 0、1 和 2（假设这个数组没有其他元素存在）。

```
$arry_name[]="mitch";
$arry_name[]="gerry";
$arry_name[]="tim";
```

2．从数组中取值

在知道了如何给数组元素赋值之后，下一步讨论一下如何获取这些值。要得到数组名为 arr_zoo，数组下标为"pelican"的值时，可以使用以下方法：

```
$key="pelican";
$value=$arr_zoo[$key];
```

运行完这两行代码以后，$value 的值将变为"Bird with a big beak,"。当然也可以用文本字符串来指定要取出哪一个数组元素的值。如：

```
$value=$arr_zoo[pelican]
```

注意：读取一个并没有赋值的数组元素时，PHP 将返回空字符串。用数组下标的字符值不应该用单引号括起来。

以上只采用了字符串作为数组下标，考虑一下下面所创建的数组初始值的数组下标是什么？

```
$arr_mixed=array(1,434,"jake"=>"23 first lane","rebecca");
```

arry_mixed 数组的数组下标分别是 0、1、jake 和 2。如果数组下标没有给定，PHP 就自动提供一个。默认的数组下标是以 0 开始的，以后当数组下标没有赋值时默认值每次加一。

可以用标量变量替换所有的数组下标（数字和字符串），并仍能获取它们原有的值。可以这样写：

```
$key=1;
echo $arr_mixed[$key];
```

以上两行将显示 434。这个例子显示 PHP 是如何按照需要，把数字数据类型转换成字符数据类型的。在以下的代码行中，数组下标被初始化成字符串：

```
$key="1";
echo $arr_mixed[$key];
```

这两行代码也显示 434，表明了 PHP 可以自动地把字符串转换成数值。

3．多维数组

对于大多数程序来说，仅有一个简单的数值列表是很不够的。例如，假如既要存储书的总页数，又要存储出版商的名字。这需要使用两个列表：lst_number_of_pages 和 lst_publisher_names，在需要增加或修改信息的时候，就很不方便了。并且在保证两个列表的同步时也留下了隐患。多维数组提供了一个极灵活的数据结构，每一个数组元素均可以包含另外一个数组。

多维数组可以按以下格式进行初始化：

```
$arr_books=arry(
 '0-679-76781-9'=>array(
     'name'=>'the Demolished Man','pages'=>243,
```

```
        'publisher'=>'vintage books'),
    ,'0-312-85395-5'=>array(
        'name'=>'Children of the Mind,'pages'=>349,
        'publisher'=>'tor books')
);
```

在上例中使用了每本书的 ISBN 号作为检索数组$arr_books 的多维数组下标，且每本书都有它自己的子数组，用以描述其自己的特定信息。为了查询子数组中的信息，正常的数组下标被扩展为采用两个下标，例如：要查《The Demolished Man》一书的页数有多少，可以用以下的表达式。

```
$arr_books[0-31-85395-5][pages]
```

用户将会发现多维数组十分灵活方便，以至于随时都可以加入需要的元素，只需要简单的在多维数组中加入如下信息。

```
$arr_books[0-312-85395-5][author]='orson scott  card'
```

多维数组数据结构允许在一个子数组中加入一项内容，而不会影响其他数据。当开始使用 PHP 从多个数据库表中收集信息时，多维数组可以用来组合这些信息。例如，当使用一个涉及多个产品供应商的产品数据库时，可以从第一个产品供应商开始将相关信息存入多维数组中。然后，再读取第二个产品供应商的信息，并将这些新信息写入同一个多维数组中。读取所有产品供应商信息，并全部写入多维数组以后，这个多维数组就包含了所有的存货信息。

5.8 函　　数

5.8.1 函数定义

在 PHP 5 中，除了使用 PHP 本身内置的内部函数外，还允许程序设计者根据需要，将一段常用的代码定义成一个函数，实现一定的功能，使程序设计模块化，让代码结构清晰易懂。

PHP 5 中的用户定义函数和 C 语言一样，包括有返回值的和无返回值的函数。函数定义的语法如下：

```
function 函数名(参数列表) {
    函数体语句
}
```

在使用时，自定义的函数名称前要加入 function 保留字，表示这是使用者自定义的函数。函数名由英文字母开头，其后是数字、字母或者下划线组成。

参数可以事先定义初始值或内定值。有定义内定值的参数在使用函数时可以省略，但一定要放在没有设定内定值参数的后面，否则 PHP 5 在解析函数时，会出现错误。另外，参数可以是 PHP 5 支持的任何变量类型，包括数组、字符串或者整数等，返回值也是一样。

函数体的语句是函数的主体，完成某一特定的功能。函数若有返回值，使用 return 语句可将值返回。

5.8.2 函数调用

定义了函数之后，可以通过调用函数，执行函数的功能。如果定义的函数有返回值，则可以把函数调用写在表达式中。如果函数没有返回值，则书写为一个语句形式。下面通过一个例子，说明函数的定义和调用过程。

【例 5.19】 求圆面积的函数定义和调用。

```
<?
function area_circle($r) {
   $area=3.14159 * $r * $r;
   return($area);
}
echo "半径为 6 的圆面积是:".area_circle(6);
?>
```

在该程序中，首先定义一个 area_circle()函数，$r 参数表示圆的半径。area_circle(6)是函数调用，表示计算半径为 6 的圆面积。

1．返回值

PHP 中的函数和 C 语言一样，包括有返回值的和无返回值的。在函数定义中可以通过可选的 return 语句返回一个函数值。返回值可以是任何类型，包括数组和对象。例 5.19 的 return 语句将$area 变量的值作为函数值返回。

如果函数需要返回多个值，可以通过返回数组的方法来实现，例如：

```
<?
function foo(){
   return array(0,1,2);
}
list($zero,$one,$two)=foo();
echo $zero.",".$one.",".$two;
?>
```

2．函数参数传递

通过参数列表可以传递信息到函数，该列表是以逗号作为分隔符的变量和常量列表。

PHP 支持按值传递参数（默认）、通过引用传递和默认参数值。在默认情况下，函数参数通过值传递，因此，即使在函数内部改变参数的值，也不会改变函数外部的值。如果希望允许函数修改它的参数值，就必须通过引用传递参数。

如果想要函数的一个参数总是通过引用传递，可以在函数定义中该参数的前面加上&符号。例如：

```
<?php
function add_some_extra(&$string) {
    $string .= 'and something extra.';
```

```
}
$str = "This is a string ";
add_some_extra($str);
echo $str;    // 输出: This is a string, and something extra.
?>
```

3．默认参数的值

函数可以定义 C++风格的标量参数默认值。

【例 5.20】 使用函数参数的默认值。

```
<?php
function makecoffee($type = "cappuccino") {
    return "Making a cup of $type.\n";
}
echo makecoffee();
echo makecoffee("espresso");
?>
```

上述片断的输出是：

```
Making a cup of cappuccino.
Making a cup of espresso.
```

默认值必须是常量表达式，不是变量、类成员或者函数调用。

5.8.3 函数和变量的作用域

1．函数参数的作用域

函数是一个封装好的模块，它接受的是形式参数，在调用函数过程中，函数外部的变量不能影响到函数内部。函数内部声明的变量同样不能影响到函数外部的变量，而且函数内部的变量在一般情况下，会随函数调用的结束而消失。因此，如果用户想让外部声明的变量作用到函数内部，或者函数内部的变量能够在函数调用结束后继续保存，这就需要使用关键字 global。

```
function one() {
  global $a;
  $a++;
}
$a=10;
one($a);
echo "a= $a <br>";
```

此函数结果将显示 a=11，表明了在 one()函数内部可以使用$a 变量。使用 global 关键字与传递引用参数相类似，因为在这两种情况下都改变函数外部的变量的值。

注意：在程序中要使用 global 关键字之前，请慎重考虑一下程序的设计思路，或许使用函数参数同样可行。在函数内部使用页面作用域的变量，并不是最好的程序设计方法，

因为函数的内部代码应该与程序中的其他部分相隔离。

2．变量作用域范围

变量作用域就是指它的有效范围。作用域问题牵涉两个相对的概念——全局（global）变量和局部（local）变量。一个变量采用什么样的作用域是看它是在什么地方生成的。通常，在用户自定义的函数里生成的变量将使用局部变量作用域，而在任何函数之外生成的变量将被看作全局变量。两者的含义是不同的。局部作用域的含义就是，当执行函数时函数内的变量才存在，一旦执行结束，这些变量就会被清除。而全局变量在文件范围内都不会被清除，但在用户自定义的函数里却是不可见的。先看一个例子。

【例 5.21】 输出“how are you.”程序代码。

```
<?php
$a = "how are you.";
function test() {
     echo $a;
}
test();
?>
```

例 5.21 中的程序不会有任何输出，因为$a 是一个全局变量。在函数 test()里是不可见的。在 test()里的$a 是一个局部变量，跟外面的$a 一点关系都没有。在一个函数内要使用全局变量，可以用以下方法之一：使用 global 声明或使用$GLOBAL 数组。

【例 5.22】 对例 5.21 程序代码的修改。

```
<?php
$a = "how are you";
function test() {
    global $a;
    echo $a;
}
test();
?>
```

该程序将输出“how are you”。程序中的第四行代码也可以换成“echo $GLOBALS["a"]”，输出的结果也是一样的。

3．静态变量

变量范围的另一个重要特性是静态变量。静态变量仅在局部函数域中存在，但当程序执行离开此作用域时，其值并不丢失。看看下面的例子：

【例 5.23】 静态变量的基本应用。

```
<?php
function Test() {
    static $a = 0;
    echo $a;
    $a++;
```

```
}
?>
```

在此例子中，每次调用 Test() 函数都会输出 $a 的值并加 1。

5.9 文件包含

利用文件包含方法可以将常用的功能写成一系列函数，存放在一个文件中，引用之后就可以调用文件中的函数了。可以使用 include()和 require()两个语句来引用文件。

1．include()

include()语句的功能在于包含并运行指定文件。寻找包含文件的顺序先是在当前工作目录相对的 include_path 下寻找，然后是当前运行脚本所在目录相对的 include_path 下寻找。假如 include_path 的当前工作目录是/www/，脚本中要 include 一个 include/a.php 并且在 a.php 文件中有一语句 include "b.php"，则寻找 b.php 的顺序先是/www/，然后是/www/include/。如果文件名以../开始，则只在当前工作目录相对的 include_path 下寻找。

当一个文件被包含时，其中所包含的代码继承了 include 所在行的变量范围。从该处开始，调用文件在该行处可用的任何变量在被调用的文件中也都可用。不过所有在包含文件中定义的函数和类都具有全局作用域。

【例 5.24】 include()语句的应用。

程序 1：文件名为 vars.php。

```
<?php
$color = 'green';
$fruit = 'apple';
?>
```

程序 2：文件名为 test.php。

```
<?php
echo "A $color $fruit";     // 输出 A
include 'vars.php';
echo "A $color $fruit";     // 输出 A green apple
?>
```

如果 include 出现于调用文件中的一个函数里，则被调用的文件中所包含的所有代码将表现得如同它们是在该函数内部定义的一样，所以它将遵循该函数的变量范围。

2．require()

require()语句的功能在于包含并运行指定文件。

require()和 include()除了怎样处理失败不同之外，在其他方面都完全一样。include()产生一个警告而 require()则导致一个致命错误。换句话说，如果想在丢失文件时停止处理页面，就用 require()。include()就不是这样，脚本会继续运行。同时也要确认设置了合适的 include_path。

【例 5.25】 require()语句的应用。

```
<?php
require 'prepend.php';
require $somefile;
require ('somefile.txt');
?>
```

3．require_once()

require_once()语句在脚本执行期间包含并运行指定文件。此行为和 require()语句类似，唯一的区别是如果该文件中的代码已经被包含了，则不会再次包含。

require_once()应用于在脚本执行期间同一个文件有可能被包含超过一次的情况下，想确保它只被包含一次以避免函数重定义，变量重新赋值等问题。

返回值和 include()相同。如果文件已被包含，本函数返回 TRUE。

5.10 利用 Dreamweaver MX 2004 编辑 PHP 程序

Dreamweaver MX 2004 是一种可视化的 Web 解决方案，能够快速进行 Web 应用程序的开发。Dreamweaver 的优势在于它不仅是优秀的所见即所得的编辑软件，同时也兼顾了 HTML 源代码编辑，可以让用户方便地在两种模式之间切换，用于对 Web 站点、Web 页和 Web 应用程序的设计、编码和开发。无论用户愿意享受手工编写 HTML 代码时的驾驭感还是偏爱在可视化编辑环境中工作。Dreamweaver 都会提供有用的工具，使用户拥有更加完美的 Web 创作体验。Dreamweaver MX 2004 还使用户可以使用服务器技术（例如 CFML、ASP.NET、ASP、JSP 和 PHP）生成由动态数据库支持的 Web 应用程序。

1．创建空白页面

当 Dreamweaver MX 2004 启动时，如果显示的是欢迎屏幕，则可以在欢迎屏幕的“创建新项目”下选择 PHP 创建一个 PHP 页面，如图 5.8 所示。



图 5.8 Dreamweaver MX 2004 欢迎界面

2．打开现有 PHP 页面

要打开现有页面可执行下列操作：选择“文件”→“打开”菜单，在弹出的对话框中选择需要打开的页面，如图 5.9 所示。

也可以在站点管理面板中，选中需要打开的页面，双击打开，如图 5.10 所示。

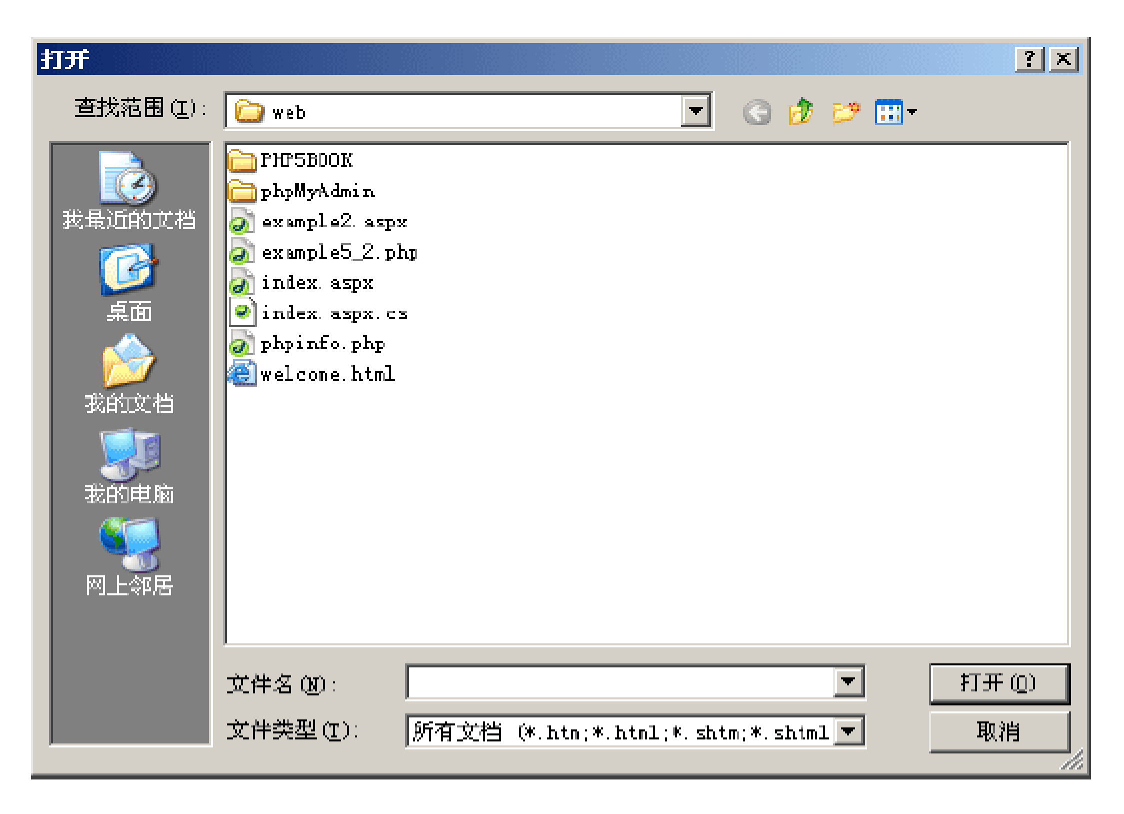





图 5.9　打开文档的界面　　　　图 5.10　站点管理面板

例如，打开 hello.php 文件，显示如图 5.11 所示的页面。接着便可利用 Dreamweaver MX 2004 的编辑工具进行源代码的修改。

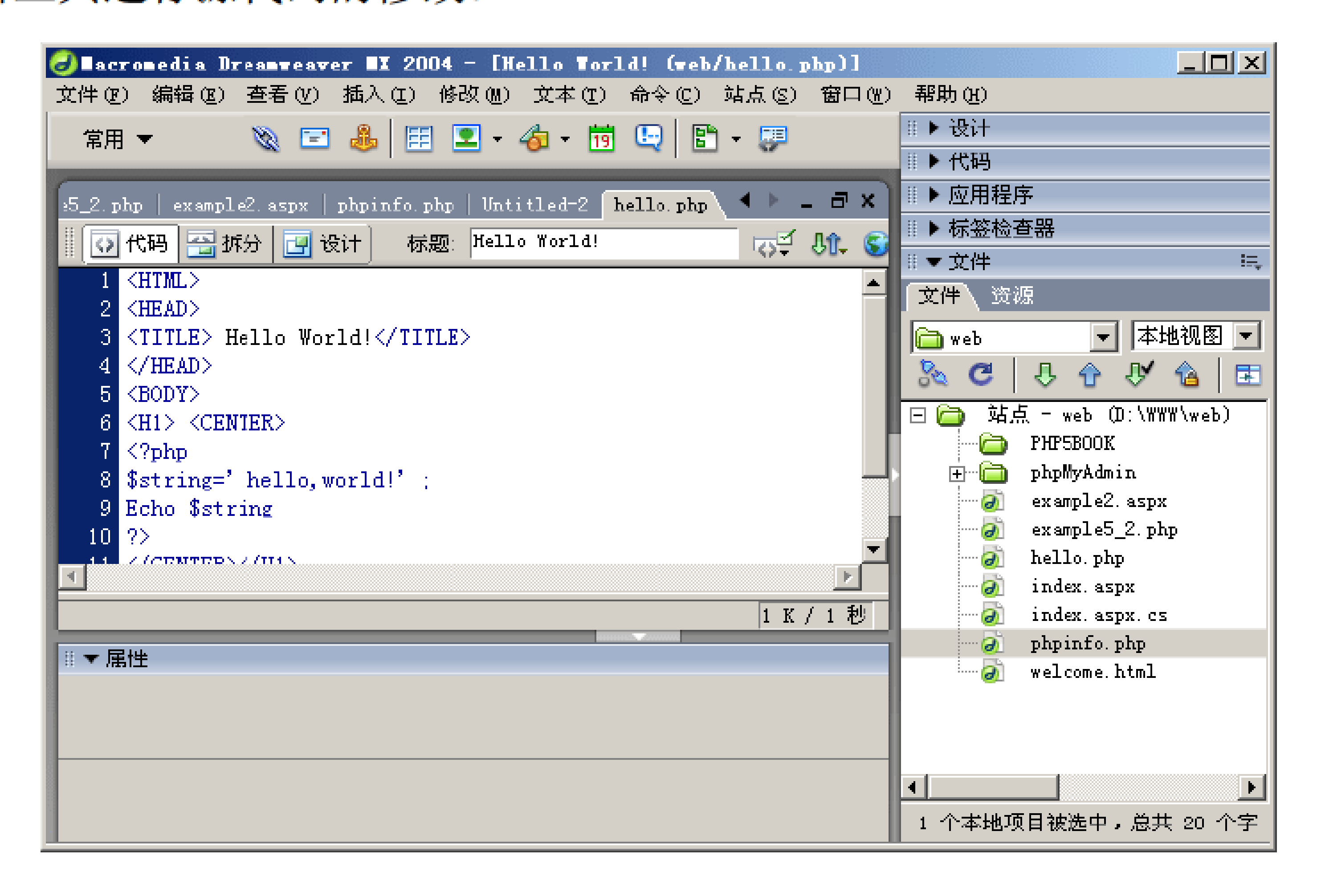


图 5.11　编辑 hello.php 程序代码的页面

3．保存文件

不管用哪种方式创建新页面，都应先将页面保存到本地站点，这样页面中的元素才能正常显示。保存页面可以按如下步骤进行：

选择“文件”→“保存”菜单，在弹出的对话框中选择要存放的站点文件夹并输入文件名，同时选择文件的类型“php 文件”，单击“保存”按钮即可。初次保存文件时，

Dreamweaver 会自动在文件后面添加扩展名 html 或 htm。文件名不能输入特殊字符、数字或中文，否则在更新文件或上传文件时会使文件链接遭到破坏。

实 验 5

1．做一个网页，提供把华氏温度转换为摄氏温度的功能。
2．随机产生一个数字作为年份，计算该年是不是闰年。

习 题 5

1．在 HTML 网页文件中嵌入 PHP 程序代码，有哪几种方式？
2．PHP 支持哪些数据类型？
3．如何利用表单来传送数据给服务器？
4．PHP 中有哪几种流程控制结构？实现分支结构和循环结构的各有哪些语句？
5．break 与 continue 语句在循环中的作用是什么？
6．PHP 函数中参数传递有哪几种方式？各有什么区别？
7．请使用 array()函数声明一个一维数组，并直接指定前三个元素的值。

第 6 章　PHP 5 的面向对象编程

本章导读

面向对象编程是目前常用的一种程序设计技术，可以极大地增强程序代码的可重用性、可扩展性与可靠性，提高应用程序的开发质量与开发速度，并减少应用程序的维护工作量。经过重新的设计与改进，PHP 5 的面向对象功能已得到了极大的增强。本章首先介绍面向对象编程的基础知识，然后详细介绍 PHP 5 中面向对象编程的基本技术与高级技术。

6.1　面向对象编程的基础知识

面向对象的编程技术（object-oriented programming，OOP）是一种与面向过程编程不同的技术，目前已得到十分广泛的应用。特别是对于各种大型应用软件的开发来说，面向对象编程更是一种首选的解决方案。

6.1.1　面向对象编程的基本概念

在面向对象编程中，应用程序的结构模块被组织为相应的对象（object）。一个面向对象的应用程序实际上就是由一系列的相关对象所构成的。作为应用程序的基本组件，对象是封装了相应属性（property）与方法（method）的实体（entity）。其中，属性描述了对象的静态特征，即对象的数据或状态；而方法则描述了对象的动态行为，即对象所能执行的功能或操作。通常，可将对象的属性理解为变量，而将对象的方法理解为函数。应用程序中各对象之间的联系是通过传递消息（message）来实现的。如果想让对象执行某个操作，那么就必须向其发送一个消息；待对象接收到消息后，便可调用相应的方法去执行指定的操作。

类（class）是面向对象编程中的一个十分重要的概念与要素。所谓类，其实就是具有相同特征与操作的一组对象的描述与定义，相当于对象的类型或分类。在一个类中，同样也封装了相应的属性与方法。通常，可将类看作是构造对象的模板或蓝本，而一个具体的对象则是相应类的一个实例。基于同一个类所生成的每一个对象，都包含有类所提供的方法，但其属性的取值却有可能不同。类和对象的关系，类似于大家所熟悉的数据类型与变量的关系，也是一种抽象与具体的关系。类的属性与方法通常又统称为类的成员。

例如，在开发一个学生成绩管理系统时，可先创建一个学生类 student。该类具有一些属性，如学号、姓名、性别等。该类也具有一些方法，如选课、退课等。有了学生类 student，便可以创建相应的学生对象，如 studentA、studentB 等。接着，便可以控制各学生对象去完成相应的操作，如选课、退课等。在此，学生类实际上是一个整体概念，可理解为所有

学生个体的统称。而每个学生对象或学生个体，则是学生类的一个具体实例。各学生对象都具有相同的属性集，但其具体取值却可能有所不同，如studentA的姓名为“卢铭”、studentB的姓名为“李兵”。另一方面，各学生对象都具有相同的方法集，通过对有关方法的调用，即可让各学生对象完成相应的操作。

6.1.2　面向对象编程的主要特征

与面向过程编程相比，面向对象编程有其明确的特征。其中，最主要的特征就是封装性（encapsulation）、继承性（inheritance）与多态性（polymorphism）。

封装性是指将数据（即属性）与操作（即方法）置于对象之中，其主要目的是为了实现对象的数据隐藏与数据保护，并为对象提供相应的接口。这样，在访问对象中的数据时，只能通过对象所提供的操作来实现。通过封装，可以有效地隐藏对象内部的具体细节，并实现对象的相对独立性，从而便于应用程序的维护与扩展。其实，封装性同样适用于类，在不同的类中即封装了该类的属性与方法。封装性是面向对象编程的主要特征之一，在某种意义上可将其看作是结构化编程技术的逻辑延伸。

继承性是指从一个已存在的类派生出另外一个或多个新类。其中，被继承的类称为父类，而通过继承所产生的类则称为子类。由于子类是从其父类继承而来的，因此子类将拥有其父类的全部属性与方法。此外，必要时还可以在子类中对所继承的属性与方法进行修改（但不能删除），或者添加新的属性与方法。更重要的是，在父类中所进行的修改会自动更新到相应的子类中。继承性是面向对象编程的重要特征，也是使应用程序具有良好的可重用性与可扩展性的根本所在。为便于理解，可将继承看作是复制类的一种特殊方式。通过继承，可以充分利用已有的程序代码，缩短应用程序的开发周期，并提高应用程序的开发质量。实际上，继承可分为两种类型，即单重继承与多重继承。其中，单重继承是指一个子类只能有一个父类，多重继承是指一个子类可以有多于一个的父类。

多态性是指同名方法（或函数）的功能可随对象类型或参数定义的不同而有所不同。实现多态性的主要方法是重载，即对类中已有的方法进行重新定义。对于某一类对象来说，在调用多态方法时所传递的参数或参数个数不同，该方法所实现的功能或过程也会有所不同。多态性也是面向对象编程的一个重要特征，一方面可以使各类对象的处理趋向一致，另一方面也有利于提高应用程序的灵活性。

6.2　PHP 5中面向对象编程的基本技术

在PHP中使用面向对象的方式进行编程时，需要先创建相应的类，然后再以类为基础创建所需要的对象，接着才能进一步访问对象的有关属性与方法。

6.2.1　类的创建

在 PHP 中，为创建一个类，需要使用关键字 class。最简单的类就是不包含任意属性与方法的空类，其创建格式为：

```
class classname
{
}
```

其中，classname 为类名，类名后的花括号“{}”则分别标识类的开始与结束。

实际上，空类是很少使用的。在创建类时，一般要根据需要在类中添加相应的属性与方法。为添加类的属性，只需使用关键字 var 声明相应的变量即可。为添加类的方法，只需使用关键字 function 定义相应的函数即可。因此，创建类的基本格式为：

```
class classname
{
   //属性
   var $propertyname_1;
   var $propertyname_2;
   ⋮
   var $propertyname_n;
   //方法
   function methodname_1(...)
   {...}
   function methodname_2(...)
   {...}
   ⋮
   function methodname_m(...)
   {...}
   }
```

其中，propertyname_1～propertyname_n 为属性名，methodname_1～methodname_m 为方法名。根据需要，各方法可以带参数，也可以不带参数。

在类的方法中，可以访问类自身的有关属性，格式为：

```
$this->propertyname
```

其中，propertyname 为属性名。在此，“$this”实际上是一个特殊的变量，用于指代当前类本身。而“->”则是 PHP 的一个运算符，用于访问类或对象的有关属性或方法。在访问类或对象的属性时，属性名之前不用带美元符“$”。

【例 6.1】 类的创建示例（class_student.php）。

```
<?php
//学生类
class student
{
  //属性
  var $xh;  //学号
  var $xm;  //姓名
  var $xb;  //性别
  //设置学生信息
  function setinfo($xh,$xm,$xb)
  {
     $this->xh=$xh;
     $this->xm=$xm;
```

```
        $this->xb=$xb;
    }
    //输出学生信息
    function getinfo()
    {
        echo "学号: $this->xh"."<BR>";
        echo "姓名: $this->xm"."<BR>";
        echo "性别: $this->xb"."<BR>";
    }
}
?>
```

在该示例所创建的学生类 student 中，共有三个属性与两个方法。其中，方法 setinfo()用于设置学生的学号、姓名与性别，方法 getinfo()则用于输出学生的学号、姓名与性别。

6.2.2 对象的使用

对象是类的实例。创建好类之后，即可为其创建相应的对象，并进一步去访问对象的属性与方法。

创建对象通常又称为实例化一个类，在 PHP 中需使用关键字 new 来实现，其基本格式为：

```
$objectname=new classname;
```

其中，objectname 为对象名，classname 为类名。

为访问对象的属性与方法，需使用“->”运算符，其基本格式为：

```
$objectname->propertyname
$objectname->methodname(...)
```

其中，objectname 为对象名，propertyname 为属性名，methodname 为方法名。应该注意的是，在调用方法时，根据其具体定义，可能需要提供相应的参数值。

对象使用完毕后，最好能及时地将其销毁，以便释放其所占用的内存空间。为此，只需将其赋为空值即可，格式为：

```
$objectname=NULL;
```

其中，objectname 为对象名，NULL 则表示空值。

【例 6.2】 对象的使用示例（student01.php）。

```
<?php
include("class_student.php");   //学生类
$MyStudent=new student;         //创建学生对象
//调用方法（设置学生信息）
$MyStudent->setinfo("200600001","卢铭","男");
$MyStudent->getinfo();     //调用方法（输出学生信息）
$MyStudent->xm="卢俊";     //访问属性（修改学生姓名）
```

```
echo "姓名: ".$MyStudent->xm;   //访问属性(输出学生姓名)
$MyStudent=NULL; //销毁对象
?>
```

在该示例中，首先通过 include 语句引入程序文件 class_student.php 的代码——学生类 student 的定义(如例 6.1 所示)，然后再创建一个学生对象 MyStudent，接着再进一步访问该对象的有关方法与属性。该示例的运行结果如图 6.1 所示。

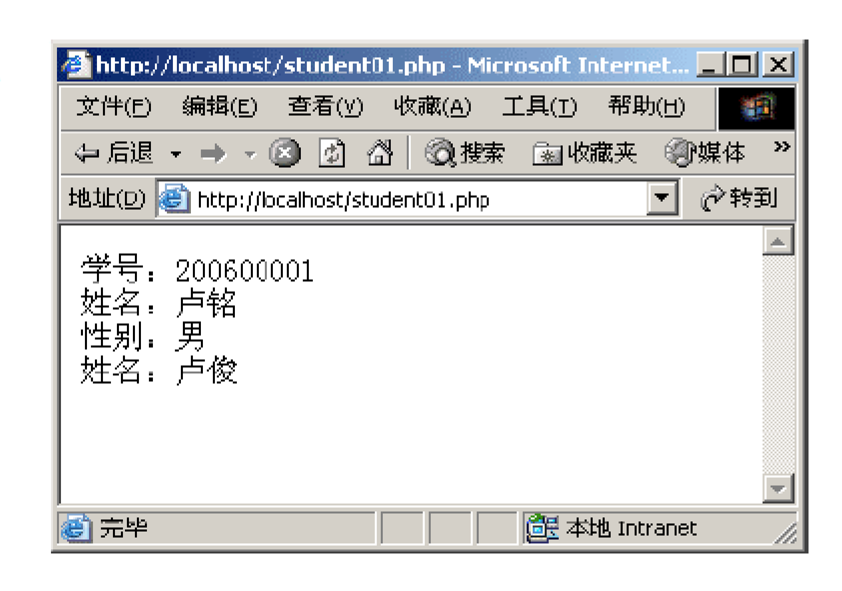


图 6.1　程序 student01.php 的运行结果

6.2.3　构造函数的使用

构造函数是类中的一个特殊函数（或特殊方法），可在创建对象时自动地加以调用。通常，可在构造函数中完成一些必要的初始化任务，如设置有关属性的初值、创建所需要的其他对象等。

在 PHP 4 及以前的版本中，构造函数的名称必须与类名相同。而在 PHP 5 中，构造函数的名称则是固定的，即必须为__construct（其中的“__”为两个下划线），而不再与类名相同。这样，当类名改变时，无须再修改构造函数的名称。与其他的函数一样，构造函数既可以带参数，也可以不带参数。

当然，为保证向下的兼容性，PHP 5 仍然允许在类中定义相应的与类名同名的方法。在这种情况下，如果没有__construct 函数，那么与类名同名的方法便是所在类的构造函数；反之，如果存在__construct 函数，那么与类名同名的方法就不是所在类的构造函数了。在 PHP 5 中创建对象时，将首先搜索有没有__construct 函数，未找到时再继续搜索有没有与类名同名的方法。

【例 6.3】 构造函数的使用示例（student02.php）。

```
<?php
//学生类
class student
{
  //属性
  var $xh;
  var $xm;
  var $xb;
  //构造函数(在此功能为设置学生的信息)
  function __construct($xh,$xm,$xb)
  {
     $this->xh=$xh;
     $this->xm=$xm;
     $this->xb=$xb;
  }
  //输出学生信息
  function getinfo()
  {
```

```
    echo "学号: $this->xh"."<BR>";
    echo "姓名: $this->xm"."<BR>";
    echo "性别: $this->xb"."<BR>";
  }
}
//创建学生对象
$MyStudent=new student("200600001","卢铭","男");
$MyStudent->getinfo();   //调用方法（输出学生信息）
$MyStudent->xm="卢俊";   //访问属性（修改学生姓名）
echo "姓名: ".$MyStudent->xm;  //访问属性（输出学生姓名）
$MyStudent=NULL;    //销毁学生对象
?>
```

在该示例中，学生类 student 的构造函数__construct() 的功能为设置学生的学号、姓名和性别（在此也可以将构造函数命名为student），实际上与例 6.1 中方法 setinfo() 的功能是一样的。由于学生类 student 定义有构造函数，因此在创建学生对象时可自动调用并完成相应的设置学生信息的功能。该示例的运行结果如图 6.2 所示。

图 6.2　程序 student02.php 的运行结果

6.2.4　析构函数的使用

与构造函数一样，析构函数也是类中的一个特殊函数（或特殊方法）。但与构造函数相反，析构函数是在销毁对象时被自动调用的。通常，可在析构函数中执行一些在销毁对象前所必须完成的操作。

在 PHP 4 及以前的版本中，是没有析构函数的。而在 PHP 5 中，则可以使用析构函数，且其名称是固定的，即必须为__destruct（其中的“__”为两个下划线）。与构造函数不同，析构函数是不能带有任何参数的。

【例 6.4】 析构函数的使用示例（student03.php）。

```
<?php
class student  //学生类
{
  var $xm;  //属性
  function __construct($xm)  //构造函数
  {
    $this->xm=$xm;
    echo "学生<".$this->xm.">来啦! <BR>";
  }
  function __destruct()  //析构函数
  {
    echo "学生<".$this->xm.">走了! <BR>";
  }
}
$MyStudent=new student("李兵");  //创建学生对象
```

```
$MyStudent=NULL;  //销毁学生对象
?>
```


在该示例的学生类 student 中，既包含有构造函数，也包含有析构函数，因此在创建与销毁学生对象时，将自动对其进行调用。该示例的运行结果如图 6.3 所示。

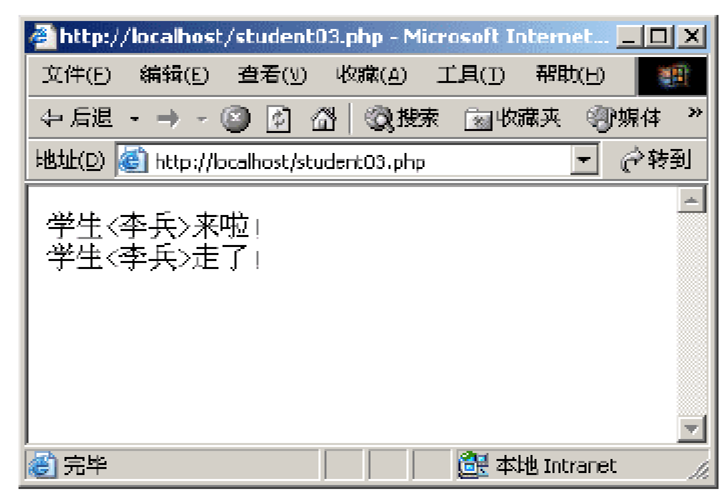


图 6.3 程序 student03.php 的运行结果

6.2.5 类属性的访问控制

使用关键字 var 所声明的属性，在类的内部与外部都可以进行访问。如果要更灵活地控制类属性的访问范围，那么可在 PHP 5 中使用新引入的访问控制关键字，即 public、private 与 protected。

关键字 public 与 var 实际上是等价的。使用关键字 public 所声明的属性，同样也是公有属性，可以在类的内部与外部进行访问，也可以被继承。其实，这也是类属性的默认访问方式。

若要缩小类属性的访问范围，可有选择地使用关键字 private 或 protected。其中，使用关键字 private 所声明的属性是私有的，只能在类的内部进行访问；而使用关键字 protected 所声明的属性则是保护的，只能在类的内部及其子类中进行访问。应该注意的是，私有属性是不能被继承的，而保护属性则可以被继承。

【例 6.5】 类属性的访问控制示例（student04.php）。

```
<?php
class student
{
  public $xh;  //学号（公有属性）
  private $xm;  //姓名（私有属性）
  protected $xb;  //性别（保护属性）
  function __construct($xh,$xm,$xb)
  {
     $this->xh=$xh;
     $this->xm=$xm;
     $this->xb=$xb;
  }
  function getinfo()
  {
     echo "学号: $this->xh"."<BR>";
     echo "姓名: $this->xm"."<BR>";
     echo "性别: $this->xb"."<BR>";
  }
}
$MyStudent=new student("200600001","卢铭","男");
$MyStudent->getinfo();
echo "学号: $MyStudent->xh"."<BR>";  //xh 为公有属性，允许访问
```

```
echo "姓名: $MyStudent->xm"."<BR>";  //xm 为私有属性, 不允许访问
echo "性别: $MyStudent->xb"."<BR>";  //xb 为保护属性, 不允许访问
$MyStudent=NULL;
?>
```

该示例的运行结果如图 6.4 所示，其中的错误是由语句“echo "姓名：$MyStudent->xm"."
";”造成的。由于 xm 为私有属性，因此试图在类的外部对其进行访问是不允许的。同样，xb 为保护属性，故也不允许在类的外部进行访问。

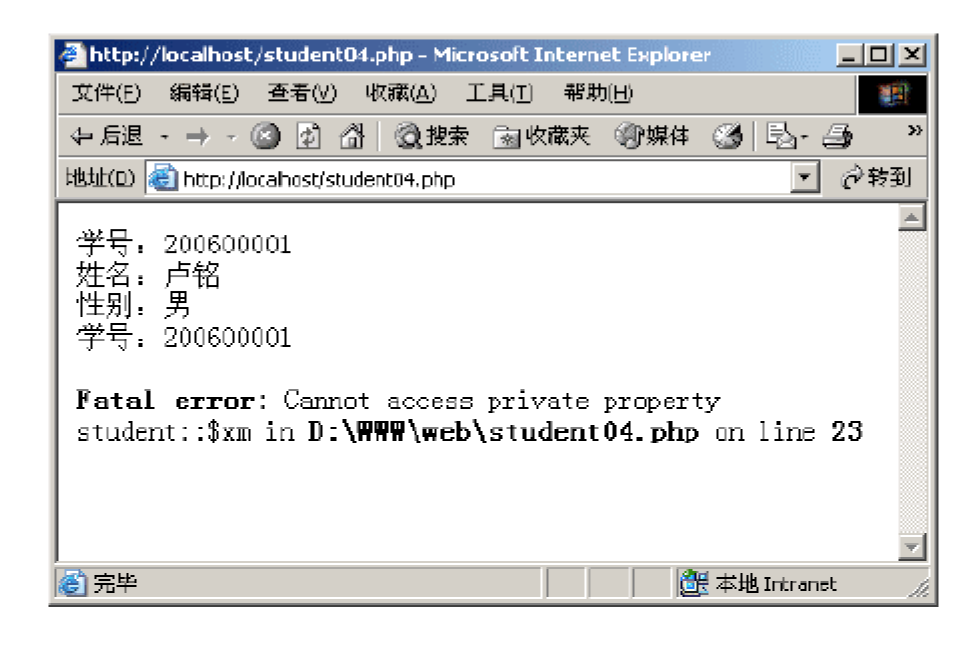


图 6.4　程序 student04.php 的运行结果

在创建类时，为更好地实现对数据的隐藏或封装，通常会将各属性声明为私有的（private）或保护的（protected）。但在实际的应用中，对属性的读取与设置操作是极其频繁的。为了有效地实现对类中各属性的访问，可在类中创建相应的_ _get()函数与_ _set()函数（其中的“_ _”为两个下划线）。_ _get()函数与_ _set()函数是 PHP 5 所提供的两个特殊函数，分别用于读取与设置类中的各个属性。其中，_ _get()函数只有一个参数，且该参数只用于传递相应属性的名称。而_ _set()函数则有两个参数，分别为相应属性的名称与所要设置的值。

【例 6.6】 _ _get()函数与_ _set()函数的使用示例（student05.php）。

```
<?php
class student
{
  public $xh;  //学号(公有属性)
  private $xm;  //姓名(私有属性)
  protected $xb;  //性别(保护属性)
  function __construct($xh,$xm,$xb)
  {
    $this->xh=$xh;
    $this->xm=$xm;
    $this->xb=$xb;
  }
  function getinfo()
  {
    echo "学号: $this->xh"."<BR>";
    echo "姓名: $this->xm"."<BR>";
    echo "性别: $this->xb"."<BR>";
  }
  function __get($propertyname)
  {
    if (isset($this->$propertyname))
      return($this->$propertyname);
    else
```

```
            return(NULL);
        }
        function __set($propertyname,$propertyvalue)
        {
          $this->$propertyname=$propertyvalue;
        }
    }
    $MyStudent=new student("200600001","卢铭","男");
    $MyStudent->getinfo();
    echo "学号: $MyStudent->xh"."<BR>";
    echo "姓名: $MyStudent->xm"."<BR>";
    echo "性别: $MyStudent->xb"."<BR>";
    $MyStudent->xh="200600002";
    $MyStudent->xm="刘莉";
    $MyStudent->xb="女";
    $MyStudent->getinfo();
    $MyStudent=NULL;
    ?>
```

该示例的运行结果如图 6.5 所示。

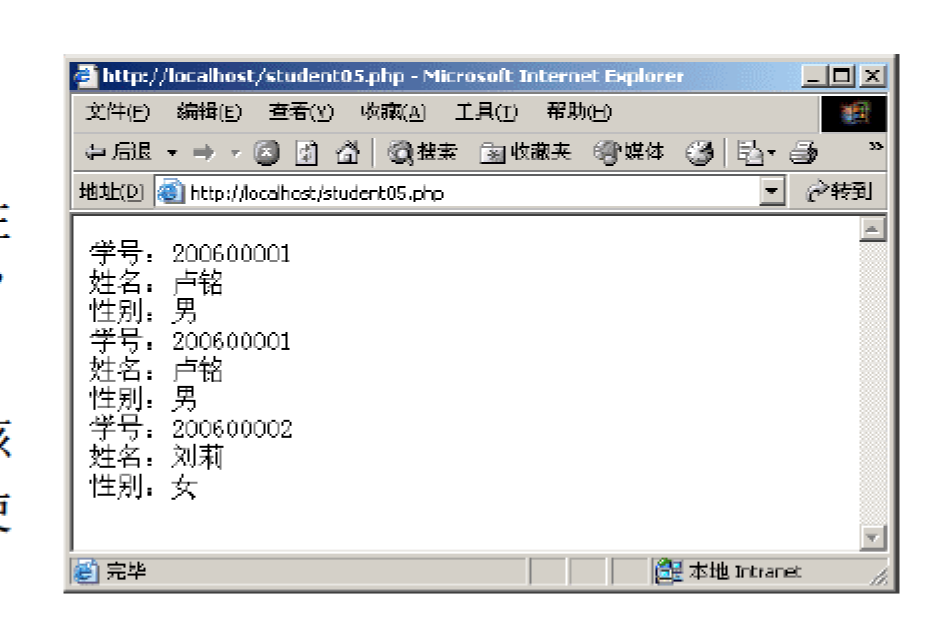


图 6.5　程序 student05.php 的运行结果

在类中所创建的__get()函数与__set()函数，可在读取与设置属性时自动地进行调用。例如，在使用语句“$myvar = $objectname->propertyname”读取属性值时，将自动调用相应的__get()函数，并将属性名 propertyname 作为该函数的参数，而该函数的返回值即为相应属性的当前值。又如，在使用语句“$objectname->propertyname = $myvalue”设置属性值时，将自动调用相应的__set()函数，并将属性名 propertyname 作为该函数的第一个参数，而将$myvalue 作为该函数的第二个参数。

在类中使用__get()函数与__set()函数来控制对属性的访问是一个值得推荐的良好的编程习惯。这样，可确保对类中各属性的访问都是通过相同的方式进行，并可在必要时进行相应的判断与处理（如空值判断、错误处理等），使程序更为健壮。

6.2.6　类方法的访问控制

在类中创建方法时，若在关键字 function 前未使用其他任何关键字，则该方法是公共的，可在类的内部与外部直接进行调用。如果要更严格地控制类方法的访问范围，那么在 PHP 5 中同样可以使用 public、private 与 protected 访问控制关键字。类方法的访问控制与类属性的访问控制是相似的。

【例 6.7】 类方法的访问控制示例（student06.php）。

```
<?php
```

```
class student
{
  public $xh;  //学号（公有属性）
  private $xm;  //姓名（私有属性）
  protected $xb;  //性别（保护属性）
  function __construct($xh,$xm,$xb)
  {
    $this->xh=$xh;
    $this->xm=$xm;
    $this->xb=$xb;
  }
  public function queryinfo()  //公有方法
  {
    $this->getinfo();
  }
  private function getinfo()  //私有方法
  {
    echo "学号: $this->xh"."<BR>";
    echo "姓名: $this->xm"."<BR>";
    echo "性别: $this->xb"."<BR>";
  }
  function __get($propertyname)
  {
    if (isset($this->$propertyname))
      return($this->$propertyname);
    else
      return(NULL);
  }
  function __set($propertyname,$propertyvalue)
  {
    $this->$propertyname=$propertyvalue;
  }
}
$MyStudent=new student("200600001","卢铭","男");
$MyStudent->queryinfo();  //queryinfo()为公有方法, 允许调用
echo "学号: $MyStudent->xh"."<BR>";
echo "姓名: $MyStudent->xm"."<BR>";
echo "性别: $MyStudent->xb"."<BR>";
$MyStudent->xh="200600002";
$MyStudent->xm="刘莉";
$MyStudent->xb="女";
$MyStudent->getinfo();  //getinfo()为私有方法, 不允许调用
$MyStudent=NULL;
?>
```

该示例的运行结果如图 6.6 所示，其中的错误是由语句“$MyStudent->getinfo();”造成的。由于 getinfo()为私有方法，因此试图在类的外部对其进行调用是不允许的。

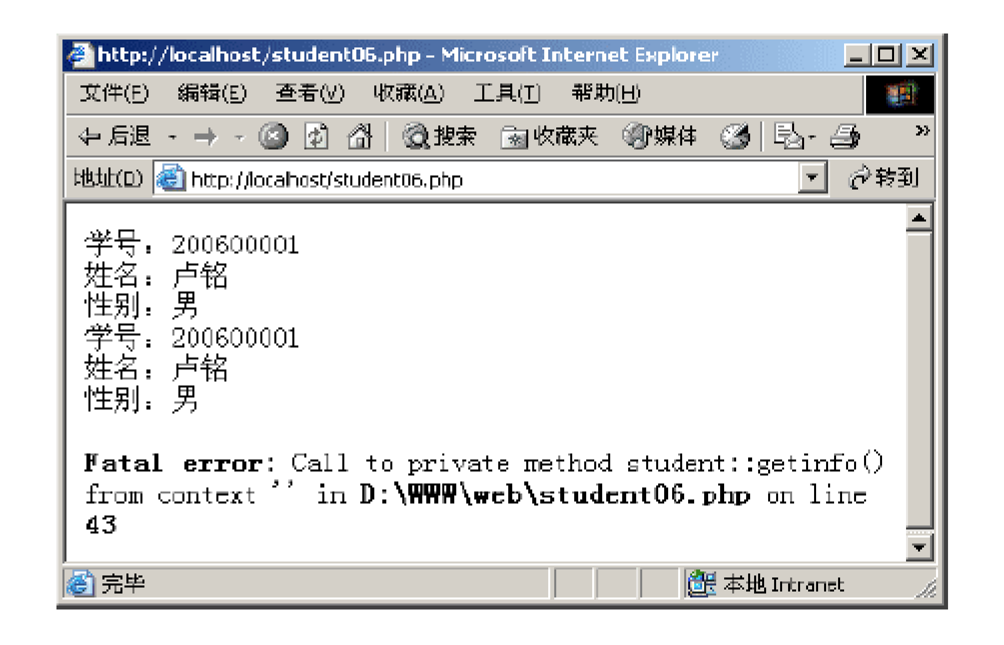


图 6.6　程序 student06.php 的运行结果

6.3　PHP 5 中面向对象编程的高级技术

为了在 PHP 中更好地使用面向对象的方式进行编程，还应掌握一些相关的高级技术，如类的继承、方法的重载、对象的克隆与串行化、静态成员的使用、抽象方法与抽象类的使用、接口的使用等。

6.3.1　类的继承

继承是面向对象编程的主要特征之一。在 PHP 中，只支持单重继承，即一个子类只能有一个父类。通过继承而生成的子类，将自动拥有父类的有关属性与方法（在父类中，被声明为 private 的属性与方法是不能被继承的，而其他属性与方法则可以被继承）。此外，还可根据需要声明相应的新属性或定义相应的新方法。必要时，也可重新声明父类中的同名属性（如改变其默认值），或重新定义父类中的同名方法（如改变其所实现的功能）。

对于 PHP 来说，类的继承是很容易实现的，只需使用关键字 extends 即可，其基本格式为：

```
class childclassname extends parentclassname
{
   //新属性
   var|public|private|protected $newpropertyname_1;
   var|public|private|protected $newpropertyname_2;
     ⋮
   var|public|private|protected $newpropertyname_n;
   //新方法
   [public|private|protected] function newmethodname_1(...)
   {...}
   [public|private|protected] function newmethodname_2(...)
   {...}
     ⋮
```

```
[public|private|protected] function newmethodname_m(...)
{...}
}
```

其中，childclassname 为子类名，parentclassname 为父类名，newpropertyname_1～newpropertyname_n 为新属性的名称，newmethodname_1～newmethodname_m 为新方法的名称。根据需要，各属性与方法均可指定为相应的访问方式。此外，各新方法可以带参数，也可以不带参数。

【例 6.8】 类的继承示例（student07.php）。

```
<?php
//学生类 student
include("class_student.php");
//学生类 student_A
class student_A extends student
{
  //新属性
  private $department;
  private $specialty;
  //新方法
  function setdepartment($department)
  {
    $this->department=$department;
  }
  function getdepartment()
  {
    echo "系别: $this->department"."<BR>";
  }
  function setspecialty($specialty)
  {
    $this->specialty=$specialty;
  }
  function getspecialty()
  {
    echo "专业: $this->specialty"."<BR>";
  }
}
$MyStudent=new student_A;
$MyStudent->setinfo("200600001","卢铭","男");
$MyStudent->setdepartment("计信系");
$MyStudent->setspecialty("计算机应用专业");
$MyStudent->getinfo();
$MyStudent->getdepartment();
$MyStudent->getspecialty();
$MyStudent=NULL;
?>
```

在该示例中，父类 student 中的所有属性与方法都是公有的（如例 6.1 所示），因此其子类 student_A 通过继承也拥有同样的属性与方法。此外，在子类 student_A 中，还另外声明了两个新属性，并为其分别定义了两个新方法。该示例的运行结果如图 6.7 所示。

在子类中，也可以访问父类中的有关方法，格式为：

```
parent::methodname(...);
```

其中，methodname 为方法名。在此，“parent”实际上是一个特殊的类，用于指代当前类的父类。而“::”则是 PHP 的作用域运算符。

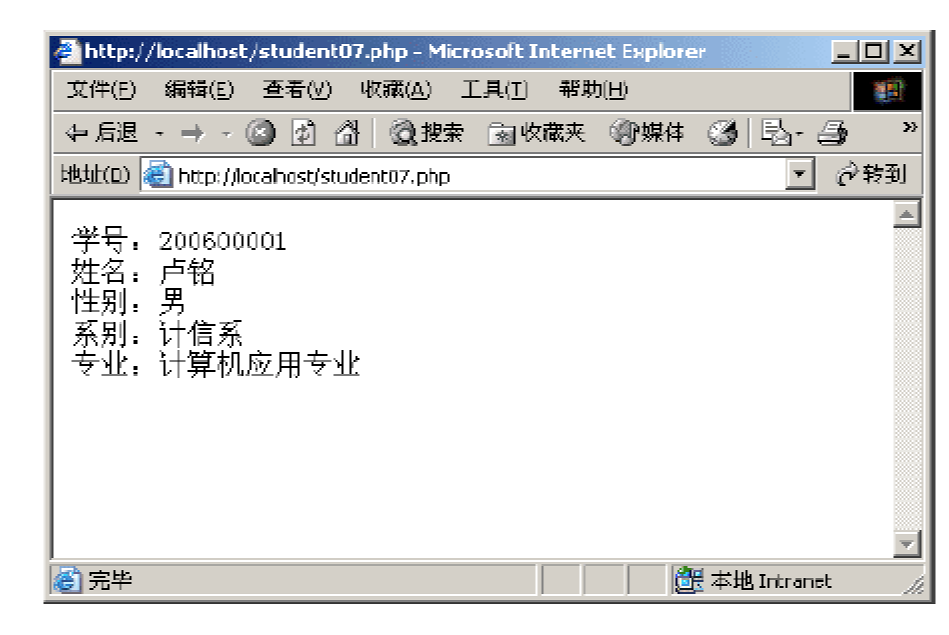


图 6.7 程序 student07.php 的运行结果

如果不想让一个类被继承，那么可在创建该类时，使用关键字 final 进行声明。

【例 6.9】 类的继承示例（student08.php）。

```
<?php
//学生类 student
include("class_student.php");
//学生类 student_A
final class student_A extends student
{
  //新属性
  private $department;
  private $specialty;
  //新方法
  function setdepartment($department)
  {
    $this->department=$department;
  }
  function setspecialty($specialty)
  {
    $this->specialty=$specialty;
  }
  function getinfo_A()
  {
    parent::getinfo();
    echo "系别: $this->department"."<BR>";
    echo "专业: $this->specialty"."<BR>";
  }
}
$MyStudent=new student_A;
$MyStudent->setinfo("200600001","卢铭","男");
$MyStudent->setdepartment("计信系");
$MyStudent->setspecialty("计算机应用专业");
```

```
$MyStudent->getinfo_A();
$MyStudent=NULL;
//学生类 student_B
class student_B extends student_A
{
}
?>
```

该示例的运行结果如图 6.8 所示，其中的错误信息是由程序中的最后三行代码产生的（子类 student_B 试图继承父类 student_A，但父类 student_A 已被声明为 final，故不允许被继承）。

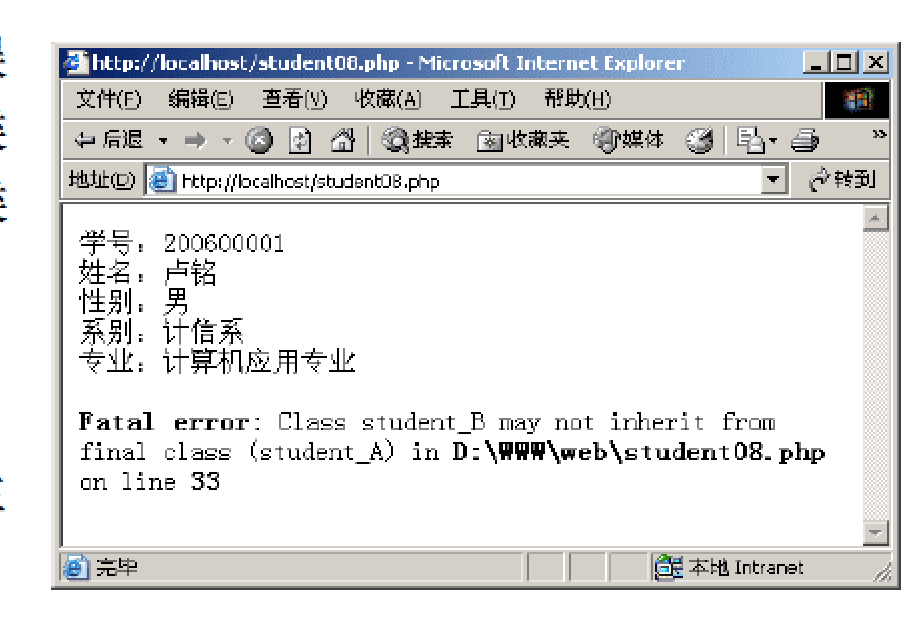


图 6.8　程序 student08.php 的运行结果

6.3.2　方法的重载

多态也是面向对象编程的主要特征之一，其主要的实现方式就是方法的重载。所谓方法的重载，是指在子类中重新定义父类中的同名方法。

【例 6.10】 方法的重载示例（student09.php）。

```
<?php
//学生类 student
include("class_student.php");
//学生类 student_A
class student_A extends student
{
  //新属性
  private $department;
  private $specialty;
  //重载父类的方法 setinfo
  function setinfo($xh,$xm,$xb,$department,$specialty)
  {
    $this->xh=$xh;
    $this->xm=$xm;
    $this->xb=$xb;
    $this->department=$department;
    $this->specialty=$specialty;
  }
  //重载父类的方法 getinfo
  function getinfo()
  {
    echo "学号: $this->xh"."<BR>";
    echo "姓名: $this->xm"."<BR>";
    echo "性别: $this->xb"."<BR>";
    echo "系别: $this->department"."<BR>";
    echo "专业: $this->specialty"."<BR>";
```

```
    }
  }
  $MyStudent=new student_A;
  $MyStudent->setinfo("200600001","卢铭","男","计信系","计算机应用专业");
  $MyStudent->getinfo();
  $MyStudent=NULL;
  ?>
```

该示例的运行结果如图 6.9 所示。

如果不想让一个方法在子类中被重载，那么可在定义该方法时，使用 final 关键字进行声明。

图 6.9　程序 student09.php 的运行结果

【例 6.11】 方法的重载示例（student10.php）。

```
<?php
//学生类 student
include("class_student.php");
//学生类 student_A
class student_A extends student
{
  //新属性
  private $department;
  private $specialty;
  //重载父类的方法 setinfo
  function setinfo($xh,$xm,$xb,$department,$specialty)
  {
     $this->xh=$xh;
     $this->xm=$xm;
     $this->xb=$xb;
     $this->department=$department;
     $this->specialty=$specialty;
  }
  //新方法
  final function getinfo_more()
  {
    parent::getinfo();
    echo "系别: $this->department"."<BR>";
    echo "专业: $this->specialty"."<BR>";
  }
}
$MyStudent=new student_A;
$MyStudent->setinfo("200600001","卢铭","男","计信系","计算机应用专业");
$MyStudent->getinfo_more();
$MyStudent=NULL;
//学生类 student_B
class student_B extends student_A
```

```
{
  function getinfo_more()
  {
     echo "学号: $this->xh"."<BR>";
     echo "姓名: $this->xm"."<BR>";
     echo "性别: $this->xb"."<BR>";
     echo "系别: $this->department"."<BR>";
     echo "专业: $this->specialty"."<BR>";
  }
}
?>
```

该示例的运行结果如图 6.10 所示，其中的错误信息是由程序中的最后一段代码产生的（子类 student_B 试图重载父类 student_A 的 getinfo_more()方法，但该方法已被声明为 final，故不允许被重载）。

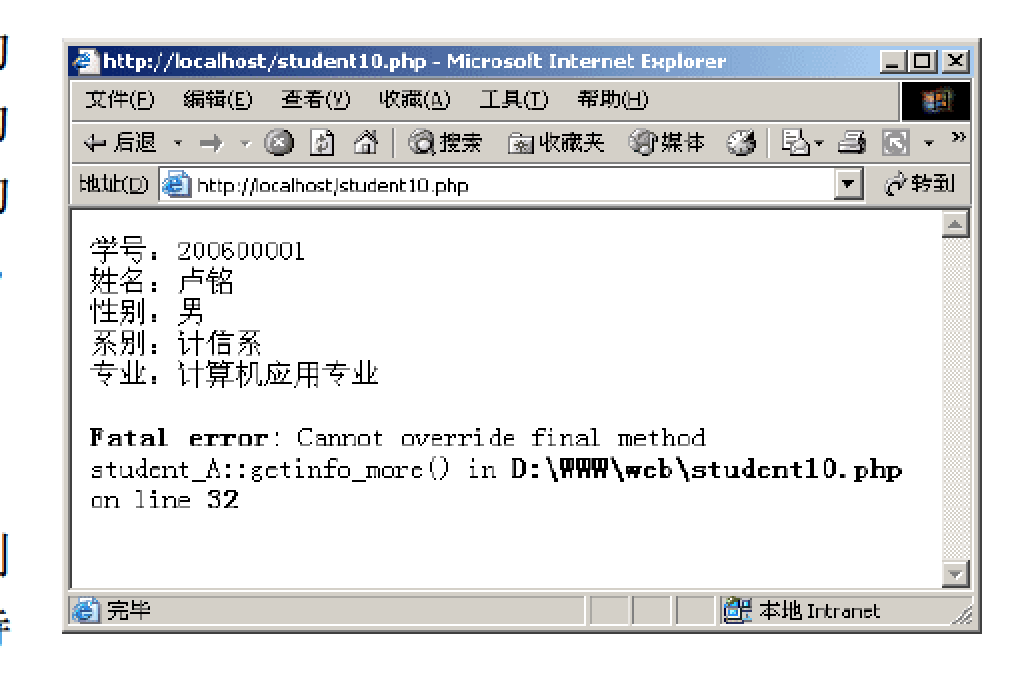


图 6.10　程序 student10.php 的运行结果

6.3.3　对象的克隆

对象的克隆是指为已存在的对象建立副本。为了实现此类应用，PHP 5 提供了一个特殊的克隆函数，且将其命名为__clone（其中的“__”为两个下划线）。

在默认情况下，在克隆对象时将建立一个与原对象具有相同属性与方法的对象。必要时，也可以在克隆对象的同时改变原对象的某些属性。为此，需要在相应类的__clone()函数中进行相应的处理。

【例 6.12】 对象的克隆示例（student11.php）。

```
<?php
class student
{
  var $xh;
  var $xm;
  var $xb;
  function __construct($xh,$xm,$xb)
  {
     $this->xh=$xh;
     $this->xm=$xm;
     $this->xb=$xb;
  }
  function getinfo()
  {
     echo "学号: $this->xh"."<BR>";
     echo "姓名: $this->xm"."<BR>";
```

```
    echo "性别: $this->xb"."<BR>";
   }
   function __clone()  //克隆函数
   {
    $this->xm=$this->xm."(克隆的)";  //修改姓名
   }
 }
 $MyStudent=new student("200600001","李兵","男");
 echo "原来的对象: <BR>";
 $MyStudent->getinfo();
 $MyStudent0=clone $MyStudent; //克隆对象
 echo "克隆的对象: <BR>";
 $MyStudent0->getinfo();
 $MyStudent=NULL;
 $MyStudent0=NULL;
 ?>
```

该示例的运行结果如图 6.11 所示。应该说明的是，对于某些版本的 PHP 5 来说，示例中克隆对象的语句“$MyStudent0 = clone $MyStudent;”应改为“$MyStudent -> __clone();”。此外，在__clone()函数中，某些版本的 PHP 5 可使用两个特殊的指针，即$this 与$that。其中，$this 指向对象的副本，$that 指向被克隆的对象。

图 6.11　程序 student11.php 的运行结果

6.3.4　对象的串行化

对象的串行化是指将对象转化为一个字符串。与此相反，对象的反串行化是指将对象的串行化字符串重新还原为原来的对象。通过对象的串行化与反串行化，即可实现对象的保存与传输等特殊应用。

在 PHP 中，为了实现对象的串行化与反串行化，需使用 serialize()与 unserialize()函数。其中，serialize()函数的参数为对象名，返回值为指定对象被串行化后的字符串；unserialize()函数的参数为某对象的串行化字符串，返回值为重新组织好的对象。

在调用 serialize()函数串行化一个对象时，PHP 将首先调用该对象的__sleep()函数（如果有的话）。__sleep()函数（其中的“__”为两个下划线）实际上是类中的一个特殊函数。该函数不接受任何参数，但要返回一个包含有该类对象中应被串行化的所有属性的数组（若无该函数，则所有属性均将被串行化）。通常，在该函数中可执行某些清除任务，如提交数据、关闭数据库连接等。

在调用 unserialize()函数通过反串行化重建对象后，PHP 将首先调用该对象的__wakeup()函数（如果有的话）。__wakeup()函数（其中的“__”为两个下划线）实际上也是类中的一个特殊函数。该函数不接受任何参数，通常可在其中完成某些初始化任务，如设置属性、重建数据库连接等。

【例 6.13】 对象的串行化与反串行化示例（student12.php）。

```
<?php
class student  //学生类
{
  var $xm;  //属性(姓名)
  var $sj;  //属性（时间）
  function __construct($xm)  //构造函数
  {
    $this->xm=$xm;
    $this->sj=date("h:m:s");  //当前时间
    echo "<".$this->sj."> 学生<".$this->xm.">来啦! <BR>";
  }
  function __destruct()  //析构函数
  {
    $this->sj=date("H:m:s");  //当前时间
    echo "<".$this->sj."> 学生<".$this->xm.">走了! <BR>";
  }
  function showinfo()  //显示对象信息
  {
   echo "姓名: ".$this->xm."<BR>";
   echo "时间: ".$this->sj."<BR>";
  }
  function __sleep()  //串行化时只包括 xm 属性
  {
    echo "<".date("H:m:s")."> 学生对象<".$this->xm.">被串行化啦! <BR>";
    $sxsz=array("xm");
    return($sxsz);
  }
  function __wakeup()  //反串行化时重新设置 sj 属性
  {
    $this->sj=date("H:m:s");
    echo "<".$this->sj."> 学生对象<".$this->xm.">被反串行化啦! <BR>";
  }
}
echo "[创建对象]<BR>";
$MyStudent=new student("李兵");
$MyStudent->showinfo();
sleep(1);
echo "[串行化对象]<BR>";
$MyObjStr=serialize($MyStudent);
sleep(1);
echo "[销毁对象]<BR>";
$MyStudent=NULL;
sleep(1);
```

```
echo "[反串行化对象]<BR>";
$MyStudent0=unserialize($MyObjStr);
$MyStudent0->showinfo();
sleep(1);
echo "[销毁对象]<BR>";
$MyStudent0=NULL
?>
```

该示例的运行结果如图 6.12 所示。

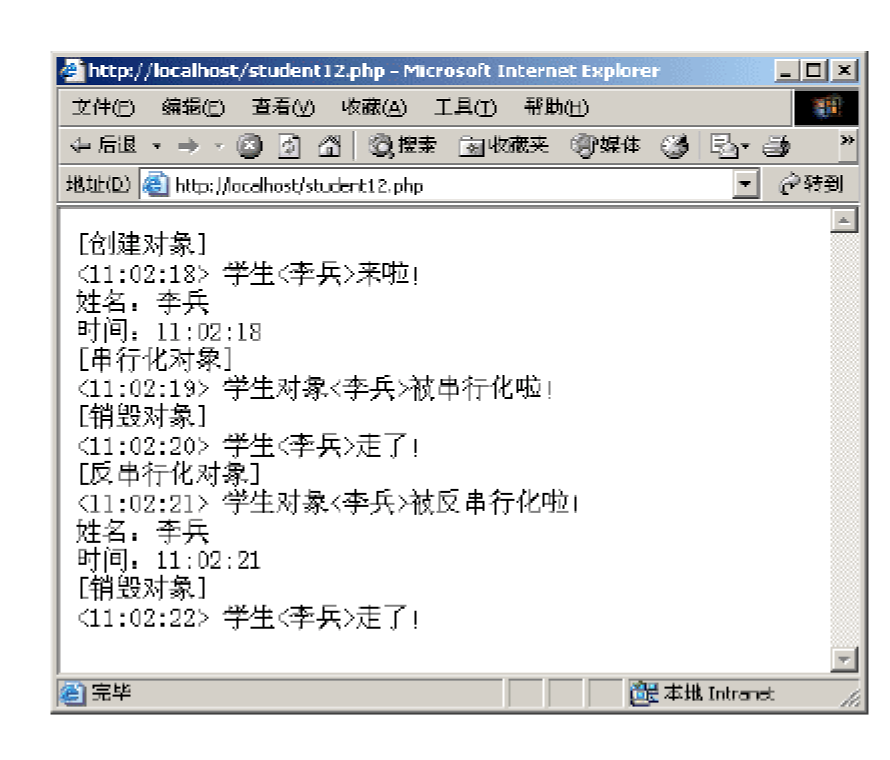


图 6.12　程序 student12.php 的运行结果

6.3.5　静态成员的使用

类的静态成员包括类的静态属性与静态方法。与一般的类成员不同，类的静态成员与对象（类的实例）无关，而只与类本身有关。事实上，静态成员所代表的是类所要封装的数据与功能，而并非特定对象所要封装的数据与功能。其中，静态成员类似于全局变量，由该类的所有实例共享；而静态方法则类似于全局函数，无须创建该类的实例即可直接进行调用。

在 PHP 中，静态成员是使用关键字 static 来进行声明的。对于静态成员，其访问方式也与一般的类成员不同。

在类的内部，静态成员应通过特殊类 self（而不是特殊变量$this）来进行访问，其基本格式为：

```
self::$propertyname
self::methodname(...)
```

其中，propertyname 为静态属性名，methodname 为静态方法名。

类似地，在类的外部，静态成员则应通过类名来进行访问，其基本格式为：

```
classname::$propertyname
classname::methodname(...)
```

其中，classname 为相应静态属性或静态方法所在类的名称。

【例 6.14】 静态成员的使用示例（student13.php）。

```
<?php
class student  //学生类
{
  private $xm;  //私有属性(姓名)
  static private $counter=0;  //静态私有属性(计数器),其初值为 0
  function __construct($xm)  //构造函数
  {
    $this->xm=$xm;
    self::$counter=self::$counter+1;
```

```
    echo "学生<".$this->xm.">来啦! 目前共有".self::$counter."人。<BR>";
  }
  function __destruct()  //析构函数
  {
    self::$counter=self::$counter-1;
    echo "学生<".$this->xm.">走了。目前共有".self::$counter."人。<BR>";
  }
  static function getcounter()  //静态方法（获取总人数）
  {
    return(self::$counter);
  }
}
echo "目前的学生总人数: ".student::getcounter()."。<BR>";
$MyStudent=new student("李兵");
$MyStudent1=new student("张强");
$MyStudent=NULL;
echo "目前的学生总人数: ".student::getcounter()."。<BR>";
$MyStudent2=new student("王志");
$MyStudent1=NULL;
$MyStudent2=NULL;
echo "目前的学生总人数: ".student::getcounter()."。<BR>";
?>
```

该示例的运行结果如图 6.13 所示。

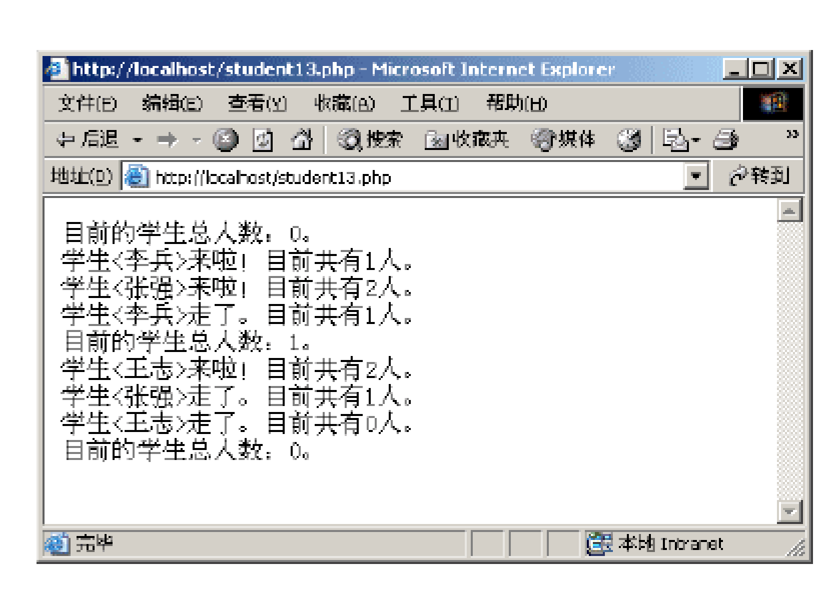


图 6.13　程序 student13.php 的运行结果

6.3.6　抽象方法与抽象类的使用

在 PHP 5 中，除了一般的类与方法以外，还可以定义和使用相应的抽象类与抽象方法。其中，抽象方法是指使用关键字 abstract 定义的尚未实现（即没有任何代码）且无任何参数的以分号“;”结束的方法，而抽象类则是指使用关键字 abstract 定义的包含有一个或多个抽象方法的类。

抽象类是不能被实例化的，但允许被继承。通过继承抽象类，可以生成相应的子类，并在其中全部或部分实现有关的抽象方法。抽象方法被实现后便成为一般的方法，而抽象类中所有的抽象方法均被实现后便成为一般的可被实例化的类。通常，可将抽象类作为其子类的模板来看待，而其所包含的抽象方法则可作为相应的一般方法的占位符来看待。

【例 6.15】 抽象方法与抽象类的使用示例（student14.php）。

```
<?php
abstract class student  //抽象类 student
{
  var $xh;
  var $xm;
  var $xb;
```

```
    //构造函数（在此功能为设置学生信息）
    function __construct($xh,$xm,$xb)
    {
      $this->xh=$xh;
      $this->xm=$xm;
      $this->xb=$xb;
    }
    //抽象方法 getinfo()
    abstract function getinfo();
  }
  class student_A extends student
  {
    function getinfo()  //实现父类中的抽象方法
    {
      echo "学号: $this->xh"."<BR>";
      echo "姓名: $this->xm"."<BR>";
      echo "性别: $this->xb"."<BR>";
    }
  }
  class student_B extends student
  {
    function getinfo()  //实现父类中的抽象方法
    {
      echo "No.: $this->xh"."<BR>";
      echo "Name: $this->xm"."<BR>";
      echo "Sex: $this->xb"."<BR>";
    }
  }
  $MyStudent_A=new student_A("200600001","卢铭","男");
  $MyStudent_A->getinfo();
  $MyStudent_A=NULL;
  $MyStudent_B=new student_B("200600001","卢铭","男");
  $MyStudent_B->getinfo();
  $MyStudent_B=NULL;
  ?>
```

该示例的运行结果如图 6.14 所示。

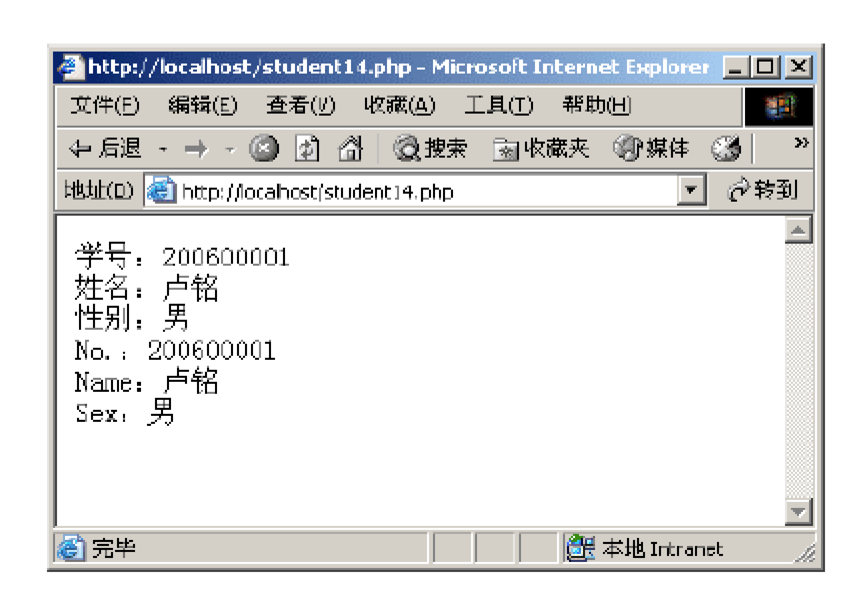


图 6.14 程序 student14.php 的运行结果

6.3.7 接口的使用

在 PHP 5 中，接口相当于一种特殊的抽象类，即只有一个抽象方法而无其他任何内容的抽象类。但与抽象类的定义不同，接口是使用关键字 interface 来进行定义的。此外，接口中方法的声明必须与其具体的实现相匹配。定义了相应的接口后，即可在创建类时使用关键字 implements 将有关的接口整合起来，并

在类中为各方法编写具体的功能代码。

【例 6.16】 接口的使用示例（student15.php）。

```
<?php
interface setdata  //接口 setdata
{
  function setinfo($a,$b,$c);
}
interface getdata  //接口 getdata
{
  function getinfo();
}
class student implements setdata,getdata  //学生类 student
{
  var $xh;
  var $xm;
  var $xb;
  function setinfo($xh,$xm,$xb)  //实现方法 setinfo
  {
     $this->xh=$xh;
     $this->xm=$xm;
     $this->xb=$xb;
  }
  function getinfo()  //实现方法 getinfo
  {
     echo "学号: $this->xh"."<BR>";
     echo "姓名: $this->xm"."<BR>";
     echo "性别: $this->xb"."<BR>";
  }
}
$MyStudent=new student;
$MyStudent->setinfo("200600001","卢铭","男");
$MyStudent->getinfo();
$MyStudent=NULL;
?>
```

该示例的运行结果如图 6.15 所示。

到目前为止，PHP 并不支持多重继承。但在 PHP 5 中所引入的接口，可看作是多重继承的一种解决方法。

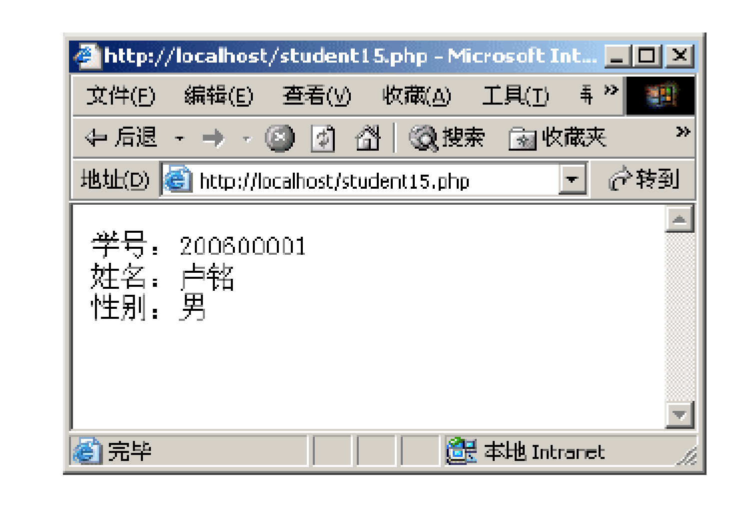


图 6.15 程序 student15.php 的运行结果

6.3.8 类方法的调用处理

当试图调用一个类中并不存在的方法时，就会产生错误。为了实现对类方法调用错误的统一处理，可使用 PHP 5 所提供__call()函数（其中的“__”为两个下划线）。

_ _call()函数是 PHP 5 所提供的一个特殊函数，可在调用不存在的方法时自动地被调用，通常用于输出相应的错误信息。该函数有两个参数，其中第一个参数用于接收相应的方法名，第二个参数用于接收相应的参数数组。

【例 6.17】 类方法的调用处理示例（student16.php）。

```
<?php
class student
{
  var $xm;
  function __construct($xm)
  {
    $this->xm=$xm;
    echo "学生<".$this->xm.">来啦! <BR>";
  }
  function __destruct()
  {
    echo "学生<".$this->xm.">走了! <BR>";
  }
  function __call($name,$args)
  {
   echo "所调用的方法".$name."()不存在! <BR>";
  }
}
$MyStudent=new student("李兵");  //创建学生对象
$MyStudent->setinfo("200600001","李兵","男");
$MyStudent->getinfo();
$MyStudent=NULL;  //销毁学生对象
?>
```

该示例的运行结果如图 6.16 所示。

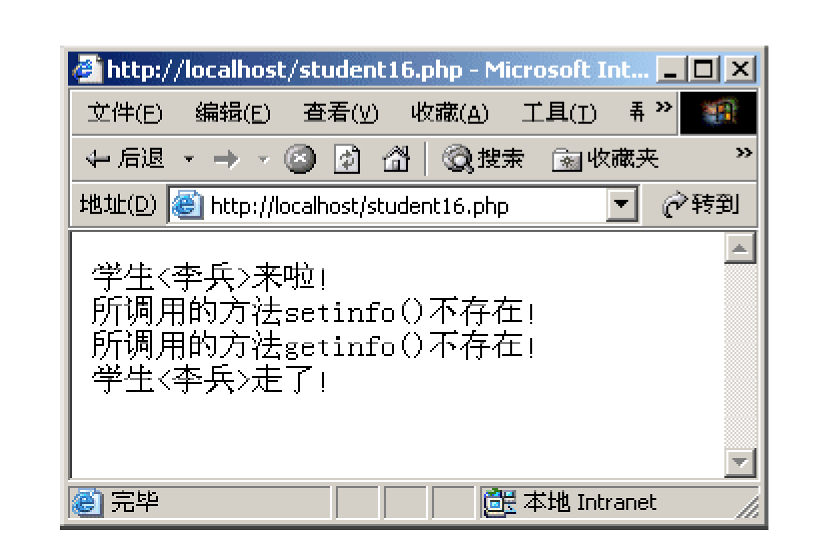


图 6.16 程序 student16.php 的运行结果

6.3.9 类文件的自动加载

在开发大型系统时，往往要创建大量的类。为便于管理、维护与使用，通常可将各个类分别保存到相应的类文件中。这样，在需要某个类时，只需通过 include 语句加载（或引入）相应的类文件即可。实际上，在 PHP 5 中，类的加载也可以由系统自动完成。为此，需使用 PHP 5 所提供_ _autoload()函数（其中的“_ _”为两个下划线）。

_ _autoload()函数是 PHP 5 的一个特殊的预定义全局函数，其功能就是自动加载所需要的类。该函数只有一个参数，用于接收由系统自动传递的类名。

【例 6.18】 类文件的自动加载示例（student17.php）。

```
<?php
```

```
//__autoload 函数
function __autoload($classname)
{
  include("class_".$classname.".php");
}
//创建学生对象
$MyStudent=new student;
//调用方法（设置学生信息）
$MyStudent->setinfo("200600001","卢铭","男");
//调用方法（输出学生信息）
$MyStudent->getinfo();
//访问属性（修改学生姓名）
$MyStudent->xm="卢俊";
//访问属性（输出学生姓名）
echo "姓名: ".$MyStudent->xm;
//销毁学生对象
$MyStudent=NULL;
?>
```

在该示例中，定义了一个__autoload()函数。当程序执行到“$MyStudent=new student;”语句时，由于找不到学生类 student，因此便将类名 student 作为参数自动调用__autoload()函数，从而引入程序文件 class_student.php 的代码——学生类 student 的定义（参见例 6.1）。该示例的运行结果如图 6.17 所示。

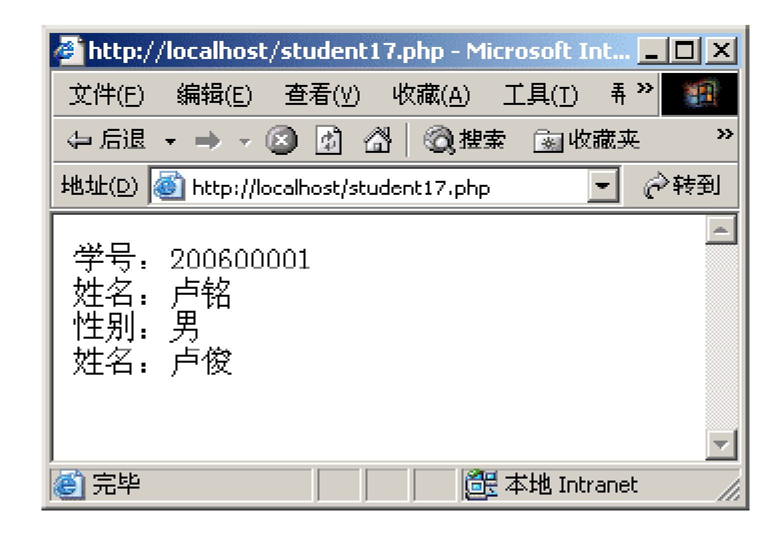


图 6.17　程序 student17.php 的运行结果

应该说明的是，类的自动加载实际上是通过自动加载类文件来实现的。因此，在命名类文件时，应遵循一定的规则，以便统一实现类的自动加载。其中，较为理想的一种命名方案就是“前缀+类名+扩展名”。例如，可将前缀指定为“class_”，而将扩展名指定为“.php”。这样，如果类名为 student 与 teacher，那么相应的类文件名就是 class_student.php 与 class_teacher.php。

实　验　6

1．请自行创建一个基本的课程类，然后再通过继承的方式创建相应的子类以扩充其功能。

2．请自行创建一个基本的购物车类，然后再通过继承的方式创建相应的子类以扩充其功能。

习　题　6

1．请简述面向对象编程的基本概念与主要特征。

2．在 PHP 中，如何创建一个类？请简述其基本格式。

3．在 PHP 中，如何创建一个对象？请简述其基本格式。

4．在 PHP 中，如何访问对象的属性与方法？请简述其基本格式。

5．请简述 PHP 中构造函数的使用方法。
6．请简述 PHP 中析构函数的使用方法。
7．请简述如何控制 PHP 中类属性的访问范围。
8．请简述如何控制 PHP 中类方法的访问范围。
9．在 PHP 中，如何实现类的继承？请简述其基本格式与使用要点。
10．在 PHP 中，如何实现方法的重载？
11．在 PHP 中，如何实现对象的克隆？
12．在 PHP 中，如何实现对象的串行化与反串行化？
13．请简述 PHP 中静态成员的使用方法。
14．请简述 PHP 中抽象方法与抽象类的使用方法。
15．请简述 PHP 中接口的使用方法。
16．请简述 PHP 5 中__call()函数的基本用法。
17．请简述 PHP 5 中__autoload()函数的基本用法。

第 7 章　常用的 PHP 5 内部函数

本章导读

为了便于程序员开发，PHP 5 函数库提供了丰富的内部函数。程序员在编写 PHP 程序时，直接利用函数库中的函数，既可以加快编写程序代码的速度，又可以方便地编写出各种各样功能强大的程序。本章详细讲述的是函数库中最常用的一些内部函数，并配有调用格式和详细解释，大部分函数还配有典型应用实例，帮助大家尽快掌握 PHP 5 内部函数的使用方法，以期达到举一反三的效果。

7.1　日期和时间函数

可以用下面这些函数得到运行 PHP 所在服务器的日期和时间，也可以用这些函数将日期和时间以多种不同方式格式化输出。

1．checkdate()函数

格式：

```
bool checkdate(int month, int day, int year);
```

该函数检查输入的年、月、日三个参数是否是有效的日期，若是返回 true；否则，返回 false。年份的值从 0～32 767，月份的值从 1～12，日期的值由月份及相应的年份决定。

【例 7.1】 checkdate()函数应用示例，文件名为 ex7_1.php。

```
<?php
echo checkdate(2,21,2006);  //返回值为 1
?>
```

因为 2006 年 2 月 21 日是一个有效的日期，所以返回 1（1 表示 true）。

2．date()函数

格式：

```
string date(string format [, int timestamp])
```

该函数格式化一个本地时间/日期，返回值是将整数 timestamp，是按照给定的 format 参数格式要求而产生的字符串。如果没有给出时间戳 timestamp，则使用本地当前时间。format 参数用来指定显示本地时间/日期的格式字符串，其值可以包括表 7.1 规定的字符和其他普通字符。

表 7.1　常用 format 参数

format 参数	说明	返回值例子
A	大写的上午和下午值	AM 或 PM
a	小写的上午和下午值	am 或 pm
D	星期中的第几天，文本表示，三个字母	Mon 到 Sun
d	月份中的第几天，有前导零的两位数字	01～31
e	时区标识（PHP 5.1.0 新加）	例如：UTC，GMT
F	月份，完整的文本格式，例如 January 或者 March	January 到 December
G	小时，24 小时格式，没有前导零	0～23
g	小时，12 小时格式，没有前导零	1～12
H	小时，24 小时格式，有前导零	00～23
h	小时，12 小时格式，有前导零	01～12
i	有前导零的分钟数	00～59
j	月份中的第几天，没有前导零	1～31
l（“L”的小写字母）	星期几，完整的文本格式	Sunday 到 Saturday
M	三个字母缩写表示的月份	Jan 到 Dec
m	数字表示的月份，有前导零	01～12
N	数字表示的星期中的第几天（PHP 5.1.0 新加）	1（表示星期一）～7（表示星期天）
n	数字表示的月份，没有前导零	1～12
S	每月天数后面的英文后缀，两个字符	st、nd、rd 或者 th
s	秒数，有前导零	00～59
T	本机所在的时区	例如，EST、MDT（如“Eastern Standard Time”，中文版会显示“中国标准时间”）
t	给定月份所应有的天数	28～31
w	星期中的第几天，数字表示	0（表示星期天）～6（表示星期六）
Y	四位数字完整表示的年份	例如，1999 或 2006
y	两位数字表示的年份	例如，99 或 06
z	年份中的第几天	0～366

说明：format 参数的格式字符串中不能被识别的字符将原样显示。

【例 7.2】 date()函数的应用范例，文件名为 ex7_2.php。

```
<?php
// 输出类似: Monday
echo date("l");
// 输出类似: Monday 15th of August 2006 03:12:46 PM
echo date('l dS \of F Y h:i:s A');
// 输出: July 1, 2000 is on a Saturday
```

```
echo "July 1, 2000 is on a ".date("l",mktime(0,0,0,7,1,2000));
?>
```

第三个语句中，mktime()是时间函数，它的一般格式为 int mktime（int hour,int minute, int second,int month,int day,int year），表示输入时间参数，则返回对应的时间戳。

在格式字串中的字符前加上反斜线来转义可以避免它被按照表 7.1 的含义解释。如果加上反斜线后的字符本身就是一个特殊序列，那还要转义反斜线，参见例 7.3。

【例 7.3】 在 date()中使用转义字符范例，文件名为 ex7_3.php。

```
<?php
  // 输出类似于 Wednesday the 15th 形式
  echo date("l \\t\h\e jS");
?>
```

此例中，字符 t、h、e 为了避免按照表 7.1 的含义解释，故需在字符前加上反斜线。但是 t 字符前加上\后为\t（是一个表示特殊含义的字符序列），因此\t 前还要加上\。

一些使用 date() 格式化日期的例子。注意，要转义所有其他字符，因为目前有特殊含义的字符会产生不需要的结果，而其余字符在 PHP 将来的版本中可能会被用上。当转义时，注意用单引号以避免类似 \n 的字符变成了换行符，参见例 7.4。

【例 7.4】 date()函数综合应用范例，文件名为 ex7_4.php。

```
<?php
// 假定今天是 March 10th, 2006, 5:16:18 pm
//下面语句结果是 March 10, 2006, 5:16 pm
$today = date("F j, Y, g:i a");
$today = date("m.d.y");         // 结果是 03.10.06
$today = date("j,n,Y");         // 结果是 10,3,2006
$today = date("Ymd");           // 结果是 20060310
//下面语句结果是 Sat Mar 10 15:16:08 MST 2006
$today = date("D M j G:i:s T Y");
//下面语句结果是 17:03:18 m is month
$today = date('H:m:s \m \i\s\ \m\o\n\t\h');
$today = date("H:i:s");         //结果是 17:16:18
?>
```

此例中的\t、\n，由于格式字符串使用了单引号，所以 date()函数没有把它们当成特殊符号来解释。

3．getdate()

格式：

```
array getdate( [int timestamp])
```

该函数获得日期 / 时间信息。返回值是一个根据 timestamp 参数得出的包含有日期信息的数组。如果没有给出时间戳则认为是当前本地时间。数组中各元素的键名及说明范例如表 7.2 所示。

表 7.2　getdate()返回的数组的键名及说明

键名	说明	返回的数组元素值例子
"seconds"	秒的数字表示	0～59
"minutes"	分钟的数字表示	0～59
"hours"	小时的数字表示	0～23
"mday"	月份中第几天的数字表示	1～31
"wday"	星期中第几天的数字表示	0（表示星期天）～6（表示星期六）
"mon"	月份的数字表示	1～12
"year"	4 位数字表示的完整年份	例如，1999 或 2003
"yday"	一年中第几天的数字表示	0～365
"weekday"	星期几的完整文本表示	Sunday 到 Saturday
"month"	月份的完整文本表示	January 到 December
0	自从 UNIX 纪元开始至今的秒数。和 time()的返回值以及用于 date()的值类似	系统相关，典型值为从–2 147 483 648～2 147 483 647

【例 7.5】 getdate()函数应用范例，文件名为 ex7_5.php。

```
<?php
$today = getdate();
print_r($today);
?>
```

上例的输出类似于：

```
Array
(
    [seconds] => 27
    [minutes] => 2
    [hours] => 9
    [mday] => 11
    [wday] => 0
    [mon] => 2
    [year] => 2007
    [yday] => 41
    [weekday] => Sunday
    [month] => February
    [0] => 1171184547
)
```

4．time()函数

格式：

```
int time()
```

返回当前的 UNIX 时间戳，其值是自从 UNIX 纪元(格林尼治时间 1970 年 1 月 1 日 00:00:00)到当前时间的秒数。

【例 7.6】 time()函数应用范例，文件名为 ex7_6.php。

```
<?php
//下一周同一时间的时间戳
$nextWeek = time()+(7*24*60*60);
echo 'Now: '.date('Y-m-d')."\n";
echo 'Next Week: '.date('Y-m-d',$nextWeek)."\n";
?>
```

上例的输出类似于：

```
Now: 2007-02-11
Next Week: 2007-02-18
```

7.2 文件操作函数

文件操作函数主要完成目录和文件操作。主要的文件操作函数有 fopen()、fclose()、feof()、fgetc()、fgets()、fputs()、fseek()、ftell()、mkdir()、readlink()、rename()、rewind()、rmdir()、unlink()等。下面就一些常用的文件操作函数进行详细介绍。

1．fopen()函数

格式：

```
resource fopen(string filename,string mode[,bool use_include_path[, resource
context]])
```

该函数打开一个文件或者 URL。如果打开文件成功，返回值是一个表示该文件或 URL 网页资源的指针；如果打开失败，本函数返回 FALSE。

参数说明：filename 参数为要打开的文件名或 URL。如果 filename 是“scheme://...”（这里的 scheme 为协议名称）的格式，则被当成一个 URL，PHP 将打开指定 URL 的网页资源。

mode 参数指定文件的访问方式，访问方式有以下几种：

- r 只读，文件指针指到文件开始处。
- r+ 可读写，文件指针指到文件开始处。
- w 写入，文件指针指到文件开始处，并将原文件的长度设为 0。若文件不存在，则建立新文件。
- w+ 可读写，文件指针指到文件开始处，并将原文件的长度设为 0。若文件不存在，则建立新文件。
- a 写入，文件指针指到文件最后。若文件不存在，则建立新文件。
- a+ 可读写，文件指针指到文件最后。若文件不存在，则建立新文件。
- b 二进制模式。

如果需要在 PHP 配置文件中的 include_path 配置选项所指定的路径中查找文件，可以将可选的第三个参数 use_include_path 设为 1 或 TRUE。

说明：对 context 的支持是 PHP 5 添加的。

【例 7.7】 fopen()函数应用范例，文件名为 ex7_7.php。

```
<?php
```

```
$handle=fopen("/home/file.txt", "r");  //UNIX 系统用法
$handle=fopen("C: \\home\\file.gif", "wb");  //Windows 系统用法
$handle=fopen("http://www.example.com/","r"); //URL 的使用范例
//URL 的使用范例
$handle=fopen("ftp://user:password@example.com/","w");
?>
```

2．fclose()函数

格式：

```
bool fclose(resource handle)
```

该函数关闭一个已打开的文件，handle 参数为要关闭的文件指针。如果成功则返回 TRUE，失败则返回 FALSE。

【例 7.8】 fclose()函数应用范例，文件名为 ex7_8.php。

```
<?php
  $handle = fopen('somefile.txt', 'r');
  fclose($handle);
?>
```

3．fgetc()函数

格式：

```
string fgetc(resource handle )
```

该函数从文件指针所指向的文件中读取一个字符，返回一个包含有一个字符的字符串。如果读取的是文件结束符 EOF，则返回 FALSE。

【例 7.9】 fgetc()例子，文件名为 ex7_9.php。

```
<?php
   $fp = fopen('somefile.txt', 'r');  //以只读方式打开文件
   if (!$fp) {                  //如果打开文件失败
       echo 'Could not open file somefile.txt';
   }
   //反复从文件中读字符并输出，直到文件内容读完为止
   while (false != ($char = fgetc($fp))) {
     echo "$char\n";
   }
?>
```

4．fgets()函数

格式：

```
string fgets(int handle [,int length])
```

该函数从文件指针 handle 参数指向的文件中读取一行，并返回长度最多为 length–1 字节的字符串。如果读取到换行符（包括在返回值中）、EOF 或者已经读取了 length–1 字节后停

止；如果没有指定 length，则默认为 1024 字节；出错时返回 FALSE。以下是一个简单例子。

【例 7.10】 fgets()函数应用范例，文件名为 ex7_10.php。

```
<?php
   $handle = @fopen("/tmp/inputfile.txt", "r");
   if ($handle) {
    while (!feof($handle)) {  //如果文件内容没有读完，则逐行读取文件
         $buffer = fgets($fd, 4096);
         echo $buffer;
    }
   fclose($handle);
  }
?>
```

说明：如果没有指定 length 参数，则默认读取一行的长度为 1024 字节。如果文件中的大多数行都大于 8KB，则在脚本中指定最大行的长度在利用资源上更为有效。

5．fgetss()函数

格式：

```
string fgetss(resource handle [,int length [,string allowable_tags]])
```

该函数从文件中读取一行并过滤掉 HTML 和 PHP 标记。它和 fgets()相同，只除了 fgetss 尝试从读取的文本中去掉任何 HTML 和 PHP 标记。可以用可选的第三个参数指定哪些标记不被去掉。

6．file()函数

格式：

```
array file(string filename [,int use_include_path [, resource context]] )
```

该函数把整个文件读入一个数组中。将文件内容作为一个数组返回。数组中的每个元素存放文件中相应的一行，包括换行符在内。如果失败，则返回 FALSE。

filename 参数是文件名，use_include_path 参数同 fopen()函数。

【例 7.11】 file()函数应用范例，文件名为 ex7_11.php。

```
<?php
  // 将一个文件读入数组，本例中通过 HTTP 从 URL 中取得 HTML 源文件
  $lines = file("http://www.example.com/");
  // 在数组中循环，显示 HTML 的源文件并加上行号
  foreach ($lines as $line_num => $line) {
   echo "Line #<b>{$line_num}</b> : ".htmlspecialchars($line)."<br/>\n";
  }
?>
```

其中，htmlspecialchars()函数的作用是把字符串中的 HTML 标记当作一般字符。因此变量 $line 代表的源文件的一行内容包括 HTML 标记全部显示出来。

7．fread()函数

格式：

```
string fread(int handle, int length)
```

该函数从文件指针 handle 所指向的文件中读取最多 length 字节。该函数在读取完 length 个字节数或到达 EOF 的时候，就会停止读取文件。

【例 7.12】 fread()函数应用范例，文件名为 ex7_12.php。

```
<?php
  // 从文件中读出内容并保存在字符串中
  $filename = "/usr/local/xu/something.txt";
  $handle = fopen($filename, "r");
  $contents = fread($handle, filesize($filename));
  echo $contents;
  fclose($handle);
?>
```

其中，函数 filesize()的作用是求指定文件的大小。

8．fwrite()函数

格式：

```
int fwrite(resource handle,string str [,int length])
```

该函数把 str 参数的内容写入文件指针 handle 所指向的文件。如果指定了 length，当写入了 length 个字节或者写完了 str 以后，写入就会停止。fwrite() 返回写入的字符数。如果写入错误，则返回 FALSE。

【例 7.13】 fwrite()函数应用范例，文件名为 ex7_13.php。

```
<?php
  $filename = 'test.txt';
  $somecontent = "添加这些文字到文件\n";
  // 首先确定文件存在并且可写，is_writable()判定文件是否可写
  if (is_writable($filename)) {
      // 使用添加模式打开$filename，文件指针将会在文件的最后
      if (!$handle = fopen($filename, 'a')) {
           echo "不能打开文件 $filename";
           exit;
      }
      // 将$somecontent 写入到打开的文件中
      if (fwrite($handle, $somecontent) == FALSE) {
         echo "不能写入到文件 $filename";
         exit;
      }
      echo "成功地将 $somecontent 写入到文件$filename";
      fclose($handle);
    } else {
    echo "文件 $filename 不可写";
  }
?>
```

9．文件通用操作函数

1）copy()函数

格式：

```
bool copy(string sourcefile,string destfile)
```

该函数将文件 sourcefile 的内容复制到文件 destfile。如果成功则返回 TRUE，失败则返回 FALSE。sourcefile 参数和 destfile 参数都是文件名字符串。

【例 7.14】 copy()函数应用范例，文件名为 ex7_14.php。

```
<?php
  $file = "example.txt ";
  $newfile = "example.txt.bak ";
  if (!copy($file, $newfile)) {
     echo "failed to copy $file...\n";
  }
?>
```

2）unlink()函数

格式：

```
bool unlink(string filename)
```

该函数删除 filename 参数指定的文件。如果成功则返回 TRUE，失败则返回 FALSE。

7.3 字符串处理函数

字符串的相关函数有许多，但除了前面介绍的 echo()、print()、printf()等函数外，还有下面几个常用的内部函数。

1．strtoupper()、strtolower()

格式：

```
string strtoupper(string str)
string strtolower(string str)
```

这两个函数用来转换英文的大小写。strtoupper()将指定的字符串 str 转换后得到全部大写的字符串，strtolower()将指定的字符串 str 转换后得到全部小写的字符串。

由于 PHP 对于字符串的要求很严格，明明是相同的一个英文字，但会因为大小写的不同而产生差异。这样规定有很多好处，如在变量的命名上，可以减少变量传递的错误，但对于字符串的判别就很伤脑筋了。假如设计一个会员登录的网页，其中需要访问者输入账号与密码才能登录，但并不是每个人都会遵守字符大小写的问题，如在 PHP 中，TOM 与 tom 这两个字符串会被认为是两个不一样的字符串，所以在进行系统登录时，会因此而导致登录失败。类似这样的例子屡见不鲜，除了密码是这样之外，其他像数据的获取，大小写如果与数据表中的数据不符也不行。

【例 7.15】 strtoupper()、strtolower()函数应用范例，文件名为 ex7_15.php。

```
<?php
$str = "Tom Had A Little Lamb and He LOVED It So";
$str1 = strtoupper($str);
echo $str1; // 结果是 TOM HAD A LITTLE LAMB AND HE LOVED IT SO
$str2 = strtolower($str);
echo $str2; // 结果是 tom had a little lamb and he loved it so
?>
```

2．trim()函数

格式：

```
string trim(string str[,string charlist])
```

该函数删除字符串的首尾空格符及指定的字符，返回值是删除首尾空格符或指定字符后的字符串。charlist 参数可以简单地列出要删除的字符，也可以用“..”指定要删除的字符的范围。

【例 7.16】 trim()函数应用范例，文件名为 ex7_16.php。

```
<?php
$text = "\t\tThese are a few words :) ...  ";
$trimmed = trim($text);  //去掉首尾空格
//执行上一条语句的结果: $trimmed = "These are a few words :) ..."
$trimmed = trim($text, " \t.");   //去掉首尾空格、"\t"和"."
//执行上一条语句的结果: $trimmed = "These are a few words :)"
$clean = trim($text, "\x00..\x1F");
// 删除变量$text 中所有 ASCII 值为 0~31 的所有字符
?>
```

注意：如果空格符是在字符串中间，那么 trim()将无法处理，如“Tom”，trim()只能将之变成“Tom”，因为字符串间的空格被 PHP 视为有效。

3．strlen()函数

格式：

```
int strlen( string str)
```

该函数求指定字符串的长度，返回值是字符串中包含的字符个数。

【例 7.17】 strlen()函数应用范例，文件名为 ex7_17.php。

```
<?php
   $str = 'abcdef';
   echo strlen($str);   //结果为 6
   $str = ' ab cd ';
   echo strlen($str); // 结果为 7
?>
```

4．substr()函数

格式：

```
string substr(string str,int start[,int length])
```

该函数求取字符串的一部分子字符串。

说明：

（1）substr()函数从字符串 str 中取出从 start 开始，长度为 length 个数的字符串。

（2）如果无参数 length，表示取到字符串的末尾。

（3）若 start 为负数，从字符串 str 的末尾倒数 start 个字节开始取。

（4）如果 length 为负数，表示取到倒数第 length 个字符。

【例 7.18】 substr()例子，文件名为 ex7_18.php。

```
<? php
echo substr("abcdef",0,1);     //结果是"a"
echo substr("abcdef",3);       //结果是"def"
echo substr("abcdef",1,3);     //结果是"bcd"
echo substr("abcdef",-2);      //结果是"ef"
echo substr("abcdef",-3,1);    //结果是"d"
echo substr("abcdef",-3,-1);   //结果是"de"
?>
```

7.4　正则表达式函数

正则表达式的用途是建立一个“匹配”，包含了字符串应出现的字符及长度。正则表达式函数可用来比较字符串的正确性以及进行相关处理的工作。

为了理解正则表达式和正则表达式函数，下面先看一些例子，如表 7.3 所示。

表 7.3　常用的正则表达式

正则表达式	符合条件	范例
[AaBbCcDd0123]	若比较字符为括号中的字符则符合	B、c、2
[A~Z]	若比较字符为大写字母则符合	C、G、Y
[0～9]	若比较字符为数字则符合	0、3、8
[a～z][0～9]	字符串中若含有如 b0、k6 等字符串则符合	c2、i9
[a～z]{3}	由匹配中字符组成的包含 3 个字符的字符串	abd、ghk
[a～z]{3，6}	由匹配中字符组成的包含 3～6 个字符的字符串	abc、abcde
[a～z]{3，}	由匹配中字符组成的包含 3 个以上字符的字符串	abefdhk
[a～z]+	由匹配中字符组成的包含 1 个以上字符的字符串	a、ab、abdf
[a～z]*	字符串为空集合或者由匹配中字符组成的字符串	a、cd、cdff

下面将详细介绍一些常用的正则表达式函数。

1．ereg()函数

格式：

```
bool ereg(string pattern, string str[, array regs])
```

该函数以区分大小写的方式在字符串 str 中寻找与给定的正则表达式 pattern 所匹配的子串。如果找到与 pattern 中圆括号内的子模式相匹配的子串，并且函数调用给出了第三个参数 regs，则匹配项将被存入 regs 数组中。$regs[1] 包含第一个左圆括号开始的子串，$regs[2] 包含第二个子串，以此类推。$regs[0] 包含整个匹配的字符串。

如果在 str 中找到 pattern 模式的匹配则返回 TRUE，如果没有找到匹配或出错则返回 FALSE。

以下代码片断接受 YYYY-MM-DD 格式的日期,然后以 DD.MM.YYYY 格式显示。

【例 7.19】 ereg()函数应用范例，文件名为 ex7_19.php。

```
<?php
$date="2007-2-12";
if (ereg("([0-9]{4})-([0-9]{1,2})-([0-9]{1,2})",$date,$regs)) {
      echo "$regs[3].$regs[2].$regs[1]";
} else {
      echo "Invalid date format: $date";
}
?>
```

在这个程序片段中，使用 if 条件语句来判断 ereg()函数的返回值，在双引号中是建立的用于比较字符串的匹配。此匹配表示在$date 字符串中找子串，子串的开头是 4 个 0～9 之间的数字，然后是“-”，接着是 1～2 个 0～9 之间的数字，最后是 1～2 个 0～9 之间的数字。若找到匹配的子串，则存入 regs 数组中。

2．ereg_replace()函数

格式：

```
string ereg_replace(string pattern, string replacement, string str)
```

该函数在 str 参数中扫描与 pattern 参数匹配的部分，并将其替换为 replacement，返回替换后的字符串。如果没有可供替换的匹配项，则返回原字符串。

【例 7.20】 ereg_replace()函数应用范例，文件名为 ex7_20.php。

```
<?php
$str = "Long long ago. There was a monkey living in the forest, He was the
king of the monkeys...";
$pattern="monkey";
$replace="tiger";
echo $str. "<br>";
//在$str 中含有"monkey"子串的地方替换为"tiger", 并输出
echo ereg_replace($pattern,$replace,$str);
?>
```

3．eregi()函数

格式：

```
bool eregi(string pattern, string str [, array regs])
```

该函数用于不区分大小写的正则表达式匹配。它和 ereg()完全相同，只除了在匹配字母字符时忽略大小写的区别。

【例 7.21】 eregi()函数应用范例，文件名为 ex7_21.php。

```
<?php
if (eregi("z", $string)) {
    echo "'$string' contains a 'z' or 'Z'!";
}
?>
```

此程序片段中用 if 语句来判断 eregi()函数返回值：判别在字符串$string 中是否含有“z”或“Z”字符。

4．eregi_replace()函数

格式：

```
string eregi_replace(string pattern, string replacement, string str)
```

该函数用于不区分大小写替换正则表达式。它和 ereg_replace()完全相同，只除了在匹配字母字符时忽略大小写的区别，故这里不作详细介绍。

5．split()函数

格式：

```
array split(string pattern, string str [, int limit])
```

该函数用正则表达式将字符串分割到数组中。返回值是一个字符串数组，每个数组元素的值是 str 经区分大小写的正则表达式 pattern 作为边界分割出的子串。如果设定了 limit 参数，则返回的数组最多包含 limit 个数组元素，而其中最后一个数组元素包含了 str 中剩余的所有部分。如果出错，则 split()函数返回 FALSE。

【例 7.22】 split()函数应用范例，文件名为 ex7_22.php。

```
<?php
//将$passwd_line 中的前四个字段分割出来，并放到列表 list 中
list($user,$pass,$uid,$gid,$extra)=split(":",$passwd_line,5);
?>
```

如果字符串中有 n 个与 pattern 匹配的项目，则返回的数组将包含 n+1 个元素。例如，如果没有找到 pattern，则会返回一个只有一个元素的数组。当然，如果 str 为空也是这样。

【例 7.23】 用 split()解析可能用斜线、点或横线分割的日期，文件名为 ex7_23.php。

```
<? php
// 分隔符可以是斜线、点或横线
$date = "04/30/1973";
list($month,$day,$year)=split('[/.-]', $date);
echo "Month: $month; Day: $day; Year: $year<br>\n";
?>
```

6．splitі()函数

格式：

```
array spliti(string pattern, string str[,int limit])
```

该函数用正则表达式不区分大小写将字符串分割到数组中。它和 split()相同，只除了在匹配字母字符时忽略大小写的区别。

7．sql_regcase()函数

格式：

```
string sql_regcase(string str)
```

该函数将字符串逐字返回大小写字母的正则表达式。返回与 str 逐字相对应的大小写字母字符串的正则表达式。返回的表达式是将 str 中的每个字母字符转换为方括号表达式，该方括号表达式包含了该字母的大小写形式。其他字符保留不变。它可用于在仅支持区分大小写正则表达式的软件中完成不区分大小写的模式匹配。

【例 7.24】 sql_regcase()函数示例，文件名为 ex7_24.php。

```
<?php
echo sql_regcase("Foo - bar.");
?>
```

输出结果为：

```
[Ff][Oo][Oo] - [Bb][Aa][Rr].
```

7.5 FTP 函数

PHP 支持 FTP 相关函数，换言之，可以用 PHP 程序去做一些 FTP 客户端可以做的事，不过前提是配置 PHP 时必须加入 enable-ftp 的选项。

1．ftp_connect()函数

格式：

```
resource  ftp_connect(string host [,int port [,int timeout]])
```

该函数建立一个新的 FTP 连接。如果成功，返回一个连接标识号（ftp stream）；失败则返回 FALSE。

host 参数为要连接的 FTP 服务器主机名，host 后面不应以斜线结尾，前面也不需要用 ftp://开头。可选参数 port 为要连接到的 FTP 服务器的端口号，如果设置为 0，则会按照默认端口 21 连接。可选参数 timeout 用来设置网络传输的超时时间限制。超时时间可以在任何时候通过函数 ftp_set_option()及 ftp_get_option()来改变或修改。

【例 7.25】 ftp_connect( )函数应用范例，文件名为 ex7_25.php。

```
<?php
$ftp_server = "ftp.example.com";
//尝试建立 ftp 连接，如不成功，则输出不能连接的信息并中断 PHP 程序
```

```
$conn_id=ftp_connect($ftp_server) or die("不能连接 ftp 服务器: $ftp_server");
?>
```

2．ftp_login()函数

格式：

```
bool  ftp_login(resource ftp_stream,string username,string password)
```

该函数用来登录 FTP 服务器。使用用户名 usename 和密码 password 登录到由 FTP 连接标识号 ftp_stream 指定的 FTP 服务器。如果成功返回 TRUE，失败则返回 FALSE。

【例 7.26】 ftp_login()函数应用范例，文件名为 ex7_26.php。

```
<?php
$ftp_server = "ftp.example.com";
$ftp_user = "foo";
$ftp_pass = "bar";
//尝试建立 ftp 连接，如不成功，则输出不能连接的信息并中断 PHP 程序
$conn_id = ftp_connect($ftp_server) or die("Couldn't connect to
$ftp_server");
//尝试登录
if (@ftp_login($conn_id, $ftp_user, $ftp_pass)) {
    echo "Connected as $ftp_user@$ftp_server\n";
} else {
    echo "Couldn't connect as $ftp_user\n";
}
?>
```

3．ftp_nlist()函数

格式：

```
array ftp_nlist(resource ftp_stream,string directory)
```

该函数返回 FTP 服务器中指定目录的文件列表。如果成功则返回给定目录 directory 下的文件名组成的数组；否则，返回 FALSE。

【例 7.27】 ftp_nlist()函数应用范例，文件名为 ex7_27.php。

```
<?php
$ftp_server = "ftp.example.com";
$ftp_user = "foo";
$ftp_pass = "bar";
// 与指定的 ftp 服务器建立连接，并登录
$conn_id = ftp_connect($ftp_server);
$login_result=ftp_login($conn_id,$ftp_username,$ftp_pass);
// 检查登录和连接情况
if ((!$conn_id) || (!$login_result)) {
     die("FTP connection has failed !");
}
```

```
// 获得根目录的文件列表
$contents = ftp_nlist($conn_id, "/");
// 输出所有的目录文件
foreach ($contents as $entry) {
    echo $entry, "<br />\n";
}
?>
```

4. ftp_systype()函数

格式：

```
string ftp_systype(resource ftp_stream)
```

该函数获得远程 FTP 服务器的系统类型。返回远程 FTP 服务器的系统类型，如果发生错误则返回 FALSE。此函数等价于在 FTP 服务器下执行 system 或 syst 指令。ftp_stream 参数为 FTP 的连接标识号。

【例 7.28】 ftp_systype() 函数应用范例，文件名为 ex7_28.php。

```
<?php
//与指定的 ftp 服务器建立连接，并登录
$ftp = ftp_connect('ftp.keliglia.com');
ftp_login($ftp, 'user', 'password');
// 得到 FTP 服务器的系统类型
if ($type = ftp_systype($ftp)) {
     echo "Keliglia is powered by $type\n";
} else {
     echo "Couldn't get the systype";
}
?>
```

本例的一种可能输出（假如 FTP 服务器的系统为 UNIX）：

```
Keliglia is powered by UNIX
```

5. ftp_get()函数

格式：

```
bool ftp_get(resource ftp_stream, string local_file, string remote_file,int
mode [, int resumepos])
```

该函数从 FTP 服务器上下载指定的文件。ftp_get()函数用来下载 FTP 服务器上由 remote_file 参数指定的文件，并保存到由 local_file 参数指定的本地文件。传送模式 mode 参数只能为 FTP_ASCII（文本模式）或 FTP_BINARY（二进制模式）之一。如果成功则返回 TRUE，失败则返回 FALSE。

需要说明的是，所谓本地文件 local_file，是对 FTP 服务器而言的，也就是 PHP 代码所在的机器，而不是用户浏览器所在的计算机。参数 resumepos 仅适用于 PHP 4.3.0 以上版本，指 remote_file 文件中指定的位置。

【例 7.29】 ftp_get()函数应用范例，文件名为 ex7_29.php。

```
<? php
// 定义变量
$local_file = 'local.zip';
$server_file = 'server.zip';
//与指定的 ftp 服务器建立连接，并登录
$conn_id = ftp_connect($ftp_server);
$login_result=ftp_login($conn_id,$ftp_username,$ftp_pass);
// 从 FTP 服务器上采用二进制模式下载文件
if (ftp_get($conn_id,$local_file,$server_file,FTP_BINARY)) {
    echo "Successfully written to $local_file\n";
} else {
    echo "There was a problem\n";
}
// 关闭 FTP 连接
ftp_close($conn_id);
?>
```

6．ftp_fget()函数

格式：

```
bool ftp_fget(resource ftp_stream, resource handle, string remote_file, int
mode [, int resumepos])
```

该函数从 FTP 服务器下载文件并保存到本地文件中。ftp_fget()函数用来下载由 remote_file 指定的文件，并写入到本地已经被打开的一个文件中。handle 参数为本地已经打开的文件句柄。传送模式参数 mode 必须是 FTP_ASCII（文本模式）或 FTP_BINARY（二进制模式）之一。如果成功则返回 TRUE，失败则返回 FALSE。

7．ftp_put()函数

格式：

```
bool ftp_put(resource ftp_stream, string remote_file, string local_file,
int mode [, int startpos])
```

该函数用来上传本地文件到 FTP 服务器。ftp_put()函数用来上传由 local_file 参数指定的文件到 FTP 服务器，上传后的位置由 remote_file 指定。传输模式参数 mode 只能为 FTP_ASCII（文本模式）或 FTP_BINARY（二进制模式）。startpos 参数指定本地文件 local_file 开始上传的位置。如果上传成功则返回 TRUE，失败则返回 FALSE。

【例 7.30】 ftp_put()函数应用范例，文件名为 ex7_30.php。

```
<? php
//本地文件$source_file 以文本模式上传到远程文件$destination_file
$upload=ftp_put($conn_id,$destination_file,$source_file,FTP_ASCII);
?>
```

8．ftp_fput()函数

格式：

```
bool ftp_fput(resource ftp_stream, string remote_file, resource handle, int
mode [, int startpos])
```

该函数用来上传一个在已经打开的文件中的数据到 FTP 服务器，参数 handle 为已打开的文件句柄，remote_file 参数为上传到服务器上的文件名。传输模式参数 mode 只能为 FTP_ASCII（文本模式）或 FTP_BINARY（二进制模式）。如果成功则返回 TRUE，失败则返回 FALSE。

【例 7.31】 ftp_fput()函数应用范例，文件名为 ex7_31.php。

```
<? php
// 打开文件
$file = 'somefile.txt';
$fp = fopen($file, 'r');
//与指定的 ftp 服务器建立连接，并登录
$conn_id = ftp_connect($ftp_server);
$login_result=ftp_login($conn_id,$ftp_username,$ftp_pass);
// 上传文件
if(ftp_fput($conn_id, $file, $fp, FTP_ASCII)) {
    echo "Successfully uploaded $file\n";
} else {
    echo "There was a problem while uploading $file\n";
}
// 关闭 FTP 服务器和文件
ftp_close($conn_id);
fclose($fp);
?>
```

9．ftp_size()函数

格式：

```
int ftp_size(resource ftp_stream, string remote_file)
```

返回 FTP 服务器中指定远程文件的大小（字节）。如果指定文件不存在或发生错误，则返回–1。有些 FTP 服务器可能不支持此特性。

【例 7.32】 ftp_size()函数应用范例，文件名为 ex7_32.php。

```
<? php
$file = 'somefile.txt';
//与指定的 ftp 服务器建立连接，并登录
$conn_id = ftp_connect($ftp_server);
$login_result=ftp_login($conn_id,$ftp_username,$ftp_pass);
//获得文件的大小
$res = ftp_size($conn_id, $file);
```

```
if ($res != -1) {
    echo "size of $file is $res bytes";
} else {
    echo "couldn't get the size";
}
ftp_close($conn_id);
?>
```

10．ftp_mdtm()函数

格式：

```
int ftp_mdtm(resource ftp_stream, string remote_file)
```

该函数返回 FTP 服务器中指定文件的最后修改时间，并以 UNIX 时间戳的方式返回。如果发生错误或文件不存在则返回–1。

【例 7.33】 ftp_mdtm()函数应用范例，文件名为 ex7_33.php。

```
<? php
$file = 'somefile.txt';
//与指定的 ftp 服务器建立连接，并登录
$conn_id = ftp_connect($ftp_server);
$login_result=ftp_login($conn_id,$ftp_username,$ftp_pass);
//获得 somefile.txt 文件的最后修改时间
$buff = ftp_mdtm($conn_id, $file);
if ($buff != -1) {
 // 输出 somefile.txt 文件的最后修改时间: March 26 2006 14:16:41.
    echo "$file was last modified on :".date("F d Y H:i:s.", $buff);
} else {
    echo "Couldn't get mdtime";
}
// 关闭 FTP 连接
ftp_close($conn_id);
?>
```

11．ftp_rename()函数

格式：

```
bool ftp_rename(resource ftp_stream, string from, string to)
```

该函数用来更改 FTP 服务器上指定文件的名字，即把参数 from 指定的文件或目录更名为 to 参数指定的名字。如果改名成功则返回 TRUE，失败则返回 FALSE。

12．ftp_delete()函数

格式：

```
bool ftp_delete(resource ftp_stream, string path)
```

该函数删除 FTP 服务器上的一个由参数 path 指定的文件。如果成功则返回 TRUE，失败

则返回 FALSE。

13．ftp_close()函数

格式：

```
void ftp_close(resource ftp_stream)
```

该函数关闭一个活动的 FTP 连接并释放所占用的资源。使用该函数后将不能再使用当前的 FTP 连接，如果要进行相关操作，必须再次使用 ftp_connect()函数来建立一个新的连接。

ftp_quit()函数是 ftp_close()函数的别名。

【例 7.34】 ftp 函数应用的综合范例，文件名为 ex7_34.php。

```
<?
$cmd=ftp_connect("192.168.0.1");
ftp_login($cmd,"user","pass");
$dir=ftp_pwd($cmd);        //ftp_pwd()取得目前在 FTP 服务器上的路径
$result=ftp_rawlist($cmd,$dir); //ftp_rawlist()列出指定目录中所有文件
reset($result);       // reset()函数将数组的指针指到数组第一个元素
//循环输出数组中每个元素的下标和元素值，其中，each()返回数组中下一个元素的下标和值
while(list($k,$v)=each($result)){
    echo "$k is $v<br>";
}
echo "Server Type:" . ftp_systype($cmd);
ftp_pasv($cmd,1);    //切换主、被动传输模式
// 使用 ftp_get()下载文件
ftp_get($cmd,"my.txt","my.txt",FTP_BINARY);
// 使用 ftp_fget()从打开的文件中下载文件
$fp=fopen("/tmp/my.txt","w");
ftp_fget($cmd,Sfp,"my.txt",FTP_BINARY);
ftp_quit($cmd);
fclose($fp);
?>
```

可能的输出为：

```
1 is -rw-r-r—1 user user 11 Jul 30 18:10 my.txt
2 is -rw-r-r—1 user user 226 Jul 31 08:30 test.php
2 is -rw-r-r—1 user user 224 Jul 30 13:57 test2.php
Server Type:UNIX
```

7.6 mail 函数

电子邮件是因特网上最早开展的服务之一，目前，在许多网站的应用程序中，电子邮件也是一个非常重要的组成部分。本节将介绍通过 PHP 的程序设计，完成网站自动发送电子邮件的功能。发送电子邮件的函数是 mail()函数，格式如下：

格式：

```
bool mail(string to,string subject, string message, string
[additional_headers])
```

mail()函数自动将参数 message 指定的邮件内容发送给由参数 to 指定的接收者。在 to 参数中用逗号或空格分隔每个电子邮件地址，就可以指定多个收件人。Subject 表示邮件的主题，选项参数 additional_headers 可省略，表示额外的头信息。发送 mail 的路径设置在 php.ini 配置文件的 sendmail_path 选项中。

【例 7.35】 mail()函数应用范例，文件名为 ex7_35.php。

```
<?
  $result=mail("tom@163.com","My subject","Line 1\nLine 2\nLine 3");
  echo  "发送结果=$result";
?>
```

如果传递了第四个字符串参数，该串就会被插入到标题的末尾，这是添加额外标题的典型方法，多个额外标题要用新行（"\n"）进行分隔。mail()的返回值若不等于 0，即表示发送成功。

【例 7.36】 用额外字头发送邮件，文件名为 ex7_36.php。

```
<?
$message= " I am a student";
$result=mail("Jim@163.com","the subject",$message,
    "From:webmast@ 163.com\nReply-To:webmaster@163.com\n-Mailer:PHP/".
phpversion());
echo "发送结果=$result";
?>
```

为了能够通过网页编辑和发送邮件的相关信息，必须设计一个表单并发送表单。例 7.37 是一个可以编辑和发送邮件的网页例子。

【例 7.37】 邮件发送网页设计范例，文件名为 sendmail.php。

```
<html>
<body>
<?
   if(@$_POST["Send']){
      if($_POST["from"])
         mail($_POST["to"],$_POST["subject"],$_POST["body"],"From:
         $_POST["from"]);
      else
         mail($_POST["to"],$_POST["subject"],$_POST["body"]);
         $Msg="信件已经送出！";
     }
?>
<h2>e-mail 发送程序<hr></h2>
<form action=sendmail.php method=post>
```

```
<table border=1>
<tr><td>收件人: </td>
   <td><input type=text name=to size=40></td></tr>
<tr><td>寄件人: <br><font size=-1>(可不填)</font></td>
   <td><input type=text name=from size=40></td></tr>
<tr><td>主题: </td>
   <td><input type=text name=subject size=40></td></tr>
<tr><td>内容: </td>
   <td>
     <textarea name=body rows=8 cols=60></textarea>
     </td></tr>
</table>
<input type=submit value="发送" name="Send">
</form>
<? echo @$Msg;?>
</body>
</html>
```

运行以上程序，得到界面如图 7.1 所示。

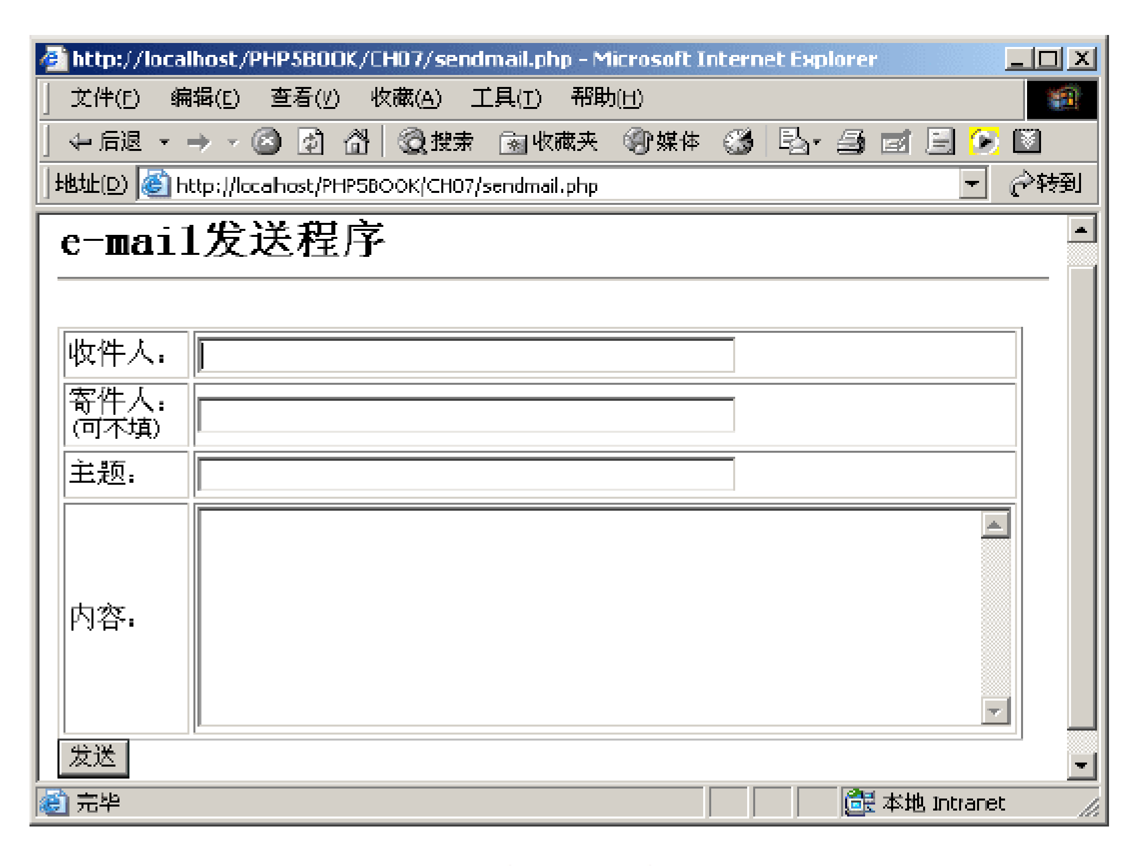


图 7.1 发送邮件的网页

实 验 7

1．请在您的网页上，利用 date()函数显示当前的日期。其样式如下：

今天是 2007 年 5 月 1 日

2．分别使用函数 date()和 getdate()显示系统日期及时间。

3．编写程序实现接受 MM.DD.YYYY 格式的日期，然后按 YYYY-MM-DD 格式显示。

4．用文件操作函数制作一个计数器程序来记录网站访问人数。

5．请按照 263.net 的免费邮箱的客户信息收集网页编制一个客户信息收集网页，在信息收集的处理程序中，需对数据是否填入进行检查，有错则向客户报错，无错则将客户信

息向客户显示以求确认。

6．设计一个表单程序，用户可以输入主机操作系统的绝对路径，使用 PHP 文件系统函数来查询和显示该路径下的所有文件。

7．假定远程 FTP 服务器上有一个文件“a.txt”，编程获得“a.txt”文件的属性，如大小、所在目录、最后修改时间等，并获得服务器系统的一些信息。

8．制作一个按月显示的日历。

习　题　7

1．用 date()函数与 getdate()函数得到系统日期的方法有什么不同？

2．写出下列程序片段的运行结果。

```
<?php
substr("abcdef",0，2);
substr("abcdef",4);
substr("abcdef",2,3);
substr("abcdef",-3);
substr("abcdef",-4,2);
substr("abcdef",-1,-4);
?>
```

3．写出下列程序片段的运行结果。

```
<?php
echo sql_regcase("Student_Abc.");
?>
```

第 8 章　MySQL 数据库基础

本章导读

MySQL 是一个真正的多用户、多线程的数据库服务器，由一个服务器守护程序及多个不同的客户程序和库组成。本章对数据库系统及 SQL 语言进行详细讲解，通过这一章的学习，将学会怎样设计和建立数据库，怎样用 SQL 语言实现数据库查询、更新等操作，怎样使用这种查询从数据表中取出信息。

8.1　MySQL 概述

8.1.1　MySQL 的概念

MySQL 是一个真正的多用户、多线程 SQL 数据库服务器。SQL（结构化查询语言）是世界上最流行的和标准化的数据库语言。MySQL 是一个客户机-服务器结构的数据库系统，它由一个服务器守护程序 mysqld 及多个不同的客户程序和库组成。

SQL 是一种标准化的语言，它使得存储、更新和存取信息更容易。例如，利用 SQL 语言为一个网站检索产品信息及存储顾客信息，同时 MySQL 足够快速和灵活以便存储记录文件和图像。

MySQL 的主要目标是快速、健壮和易用。最早是因为人们需要这样的一个 SQL 服务器，它能处理任何能提供不昂贵硬件平台上的数据库厂家的大数据库，但速度更快，MySQL 便被开发出来。

8.1.2　MySQL 数据库的特征

1．多线程

MySQL 使用的核心线程是完全的多线程。线程分成用户线程和核心线程。用户线程是通过用户级的代码来实现线程间的切换，内核并不知道一个进程内多个用户线程的存在。这种方式实际上是多个用户线程复用一个内核调度实体的运行时间，所以，即使系统中存在多个 CPU，多个用户线程也只能轮流地使用某个 CPU。而核心线程是通过核心代码来实现的，内核知道核心线程的存在，内核将每一个核心线程看成一个内核的调度实体，在这种方式下，多个核心线程可以在多个 CPU 下并发运行。MySQL 通过使用核心线程来提供更好的并发性。

2．跨平台支持

MySQL 可运行在不同的平台上，在 Linux、UNIX、Windows 平台均有 MySQL 的实现，

用户可以比较方便地实现数据库在多个平台间的移植。

3．支持丰富的数据类型

MySQL 支持丰富的数据类型，其中包括 1、2、3、4 和 8 字节长度的有符号/无符号整数、FLOAT、DOUBLE、CHAR、VARCHAR、TEXT、BLOB、DATE、TIME、YEAR、DATETIME、TIMESTAMP（时间戳）、SET（集合）和 ENUM（枚举）类型。

4．优化的查询策略

MySQL 利用一种优化的一遍扫描多重连接（one-sweep multi-join）算法，可以非常快速地进行连接（join）。MySQL 还通过一个高度优化的类库实现 SQL 函数库，这个类库可以像 sql 函数库一样快速高效地工作，并且在查询初始化后通常不需要任何内存的分配。MySQL 具备索引压缩的快速 B 树磁盘表。

5．对 ANSI SQL 的良好支持

MySQL 提供了对 ANSI SQL 的良好支持；MySQL 全面支持 SQL 的 GROUP BY 和 ORDER BY 子句，支持聚合函数 COUNT()、COUNT(DISTINCT)、AVG()、STD()、SUM()、MAX()和 MIN()；MySQL 支持 ANSI SQL 的 LEFT OUTER JOIN 和 ODBC 语法；MySQL 的表和列的别名符合 SQL 92 标准。

6．灵活安全的权限机制

MySQL 拥有非常灵活安全的访问权限和口令系统，并且它允许基于主机的认证。当与一个服务器连接时，所有的口令被加密传送，所以口令是安全的。

7．具有承载大量数据的能力

MySQL 具有大型数据库的处理能力。某些 MySQL 的应用正在尝试对包含 50 000 000 多个记录的数据使用 MySQL 系统。

8．连接的方式灵活

客户端使用 TCP/IP 连接或 UNIX 套接字（socket）或 NT 下的命名管道连接 MySQL。在 Windows95 下还提供了 ODBC for Windows 95。它支持所有的 ODBC 2.5 函数和其他许多函数。用户可以使用 Access 连接需要的 MySQL 服务器。

8.2　MySQL 的启动和关闭

MySQL 是一个基于客户机-服务器结构的数据库服务器，它由服务器程序和客户端程序组成。这两种程序的运行过程都是基于网络的，服务器程序和客户端程序既可以在同一主机上运行，也可以分别在网络中的不同主机上运行。MySQL 服务器程序文件名是以 mysqld 开头的程序文件，如 mysqld.exe、mysqld-nt.exe 等，客户端程序是 mysql.exe。要使用 MySQL 数据库系统，需要先启动 MySQL 服务器程序，然后执行 MySQL 客户端程序，连接到启动 MySQL 服务器，才能对 MySQL 数据库进行各种操作。

8.2.1　*启动 MySQL 服务器*

不管是 Windows 还是 Linux 系统平台，通过执行 MySQL 安装目录下 bin 子目录的 mysqld 程序，就可以启动 MySQL 服务器。假如在 Windows 平台下将 MySQL 安装到 D:\MySQL 目录下，则启动 MySQL 服务器的命令如下：

```
D:\MySQL\bin\mysqld.exe
```

如果是在 Windows 2000/XP 平台，也可以执行 mysqld-nt.exe，以独立进程的方式来启动 MySQL 服务器，命令如下：

```
D:\MySQL\bin\mysqld-nt.exe  --standalone
```

8.2.2 连接到 MySQL 服务器

为了在客户端能够对 MySQL 数据库进行操作，通过执行其客户端程序 mysql，连接到 MySQL 服务器。mysql 程序的命令格式如下：

```
mysql [--user=用户名] [--password=密码] [--host=主机名]
```

在该命令中，--user 选项指定 MySQL 服务器的登录用户名，--password 选项指定登录用户的密码，--host 选项指定运行 MySQL 服务器的主机名。

【例 8.1】 假设 MySQL 服务器和客户端程序都在同一主机上运行，连接到本机的 MySQL 服务器的命令如下：

```
D:\MySQL\bin\mysql
```

运行后，打开一个 DOS 窗口，显示一个提示符"mysql>"，表明当前已经连接到 MySQL 服务器，处于客户端状态，如图 8.1 所示。

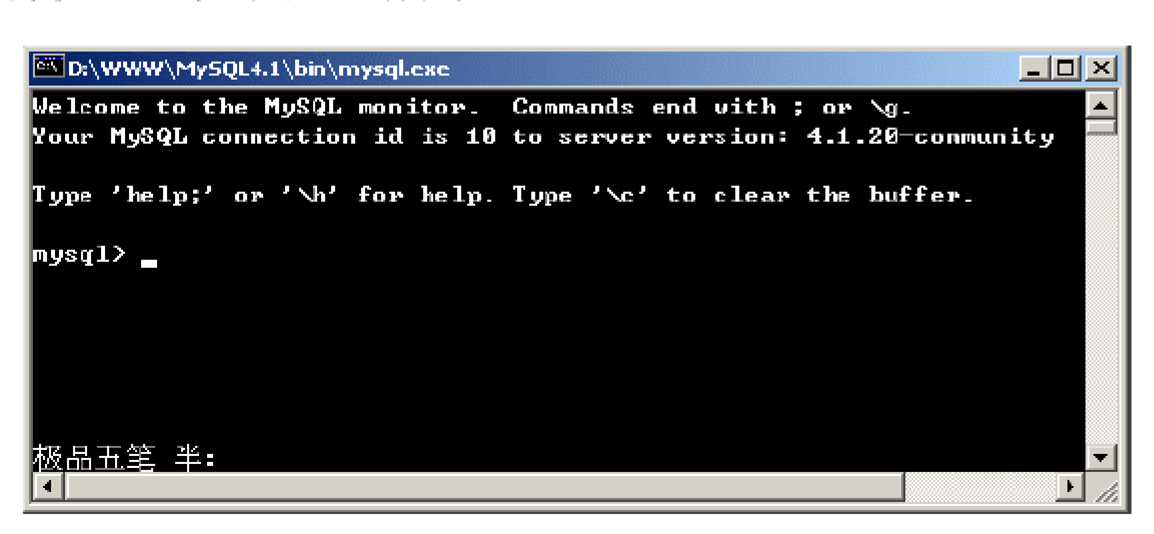


图 8.1 MySQL 客户端程序窗口

【例 8.2】 假设 MySQL 服务器运行于另一台 IP 地址为 172.16.81.93 的主机，则连接到该主机的 MySQL 服务器的命令如下：

```
mysql --user=root --password=123456 --host=172.16.81.93
```

这里，MySQL 服务器的登录用户名为 root（管理员用户），密码为 123456。

8.3 MySQL 的基本语法

8.3.1 MySQL 的命名规则

在 MySQL 中，给数据库、表、索引、列命名时，要遵守下面的规则：

（1）可以使用字母和数字以及_和$。

（2）可用数字作为开始，但不能整个命名中都是数字。

（3）在 UNIX 系统中要区分大小写，Windows 中不区分（在 Windows 最好还是统一使用大写或小写）。

（4）长度有如下的限定：数据库名、表名、列名这些标识符的最大长度是 64 个字符，别名标识符的最大长度是 255 个字符。

例如，test、customer_123、car$都是合法的命名，而 123、car*、name+test 是不合法的命名。

有一部分单词，例如，select、delete 等已经被 MySQL 作为关键字。为了保证兼容性和可移植性，建议在命名中不要使用数据库系统的关键字。

8.3.2 MySQL 的数据类型

MySQL 的数据类型分为数值型、字符型、日期时间型和 NULL 类型四种，下面将分别介绍这四种数据类型。

1．数值型

MySQL 可以处理普通十进制的数据，也可以处理十六进制的数据。在处理十六进制数据时候，可以使用“0xff”的形式，但不能使用“0Xff”的形式。如果想要显示该十六进制数所对应的十进制数，可以通过将十六进制数和十进制数进行运算的方式对十六进制数据进行转换和输出。

例如：

```
mysql>SELECT 0xf*1;
->15
mysql>SELECT 0x43434a;
->CCJ
```

2．字符型

MySQL 中的字符型数据是用来保存字符串的。在 MySQL 中的字符串用单引号或双引号引起来。例如，‘你好，PHP’、“今天是国庆节”都是合法的字符串。但是，如果在双引号括住的字符串中含有双引号，则可以采用以下两种方法来表示字符串：

方法 1：使用两个双引号，例如：

```
mysql>select "Hello,""Frank!""";
```

方法 2：使用转义字符。

```
mysql>select "Hello,\"Frank\"";
```

上述两个语句输出的结果都是“Hello,"Frank"”。

在字符串中以一条反斜线（“\”）开始的字符，称为转义字符。MySQL 能够识别下列表 8.1 中的转义字符。对于单引号的使用，与双引号类似。

3．日期和时间类型

日期和时间是一些诸如“2006-09-10”或“12:30:43”这样的值。MySQL 还支持日期和时间的组合，如“2006-09-10 12:30:43”。

注意：MySQL 是按年-月-日的顺序表示日期的。

表 8.1　MySQL 的转义字符

转义字符	含义	转义字符	含义
\0	空字符（NUL）	\r	回车符
\n	换行符	\b	退格符
\t	制表符	\\	反斜线(\)
\'	单引号(')	\"	双引号(")
_	下划线（_），它表示正文中搜索“_”符，否则“_”将被解释为一个通配符	\%	百分号（%），它表示正文中的百分号，否则“%”将被解释为一个通配符

4．NULL 值

NULL 是一种“无类型”的值。它过去常表示的意思是“无值”、“未知值”、“丢失的值”、“溢出值”以及“没有上述值”等。可将 NULL 值插入表中，从表中检索它们，测试某个值是否是 NULL，但不能对 NULL 值进行算术运算。

注意：如果对 NULL 进行算术运算，其结果为 NULL。

8.3.3　字段类型

MySQL 支持大量的字段类型，它可以被分为：数字类型、日期和时间类型以及字符串类型三类。

1．数字类型

数字类型用于存储各种数值类型的数据，它主要分为整数型和浮点型两种。所有的数字类型之后允许有两个可选属性：UNSIGNED 和 ZEROFILL。UNSIGNED 属性表示无符号数，将正数的取值范围扩大；ZEROFILL 属性表示为该数值添加前置的零，而不是空格，并且自动将它变为 UNSIGNED。表 8.2 列出了可用的数字类型。

表 8.2　MySQL 的数字类型

数字类型	描述
TINYINT[(M)]	微小整数。其有符号数范围–128~127，无符号数范围 0~255。占 1B 存储空间
BIT，BOOL	同 TINYINT(1)
SMALLINT	小整数。其有符号数范围–32 768~32 767，无符号数范围 0~65 535。占 2B 存储空间
MEDIUMINT	中等整数。其有符号数范围–8 388 608~8 388 607，无符号数范围 0~16 777 215，占 3B 存储空间
INT，INTEGER	整数。其有符号数范围–2 147 483 648~2 147 483 647，无符号数范围 0~4 294 967 295。占 4B 存储空间
BIGINT	大整数。其有符号数范围–9 223 372 036 854 775 808~9 223 372 036 854 775 807，无符号数范围 0~18 446 744 073 709 551 615，占 8B 存储空间
FLOAT(P)	浮点数。P 为精度，P≤24 用于单精度浮点数，占 4B 存储；25≤P≤53 时用于双精度浮点数，占 8B 存储空间
FLOAT[(M,D)]	单精度数。取值范围–3.402823466E+38～–1.175494351E-38，0 和 1.175494351E–38~ 3.402823466E+38，占 4B 存储空间

续表

数字类型	描述
DOUBLE[(M,D)]	双精度数。取值范围–1.7976931348623157E+308~–2.2250738585072014E–308、0 和 2.2250738585072014E–308~1.7976931348623157E+308，占 8B 存储空间
DOUBLE PRECISION	同 DOUBLE[(M,D)]
REAL	DOUBLE[(M,D)]
DECIMAL(M,D)	非压缩的定点数。将一个数像字符串那样存储，每个字符占一个 B。M 是数字的总位数（不含小数点和符号），D 是小数点后的小数位数。占用存储空间为 M+2B（D>0 时）、M+1B（D=0 时）或 D+2B（M<D 时）
NUMERIC(M,D)	同 DECIMAL(M,D)

NUMERIC 和 DECIMAL 类型被 MySQL 实现为同样的类型。它们被用于保存精度准确的值，例如与金钱有关的数据。当声明一个字段是这些类型之一时，需要指定数值的精度和小数位。例如，在“salary DECIMAL(9,2)”中，9 代表将被存储值的有效数字位数，而 2 代表该数小数点后的位数。因此，salary 列能够存储的数值范围是–9 999 999.99~9 999 999.99。

在 MySQL 中，为了保证存储数值的小数精度，DECIMAL 等数值是以字符串形式存储的，数值中的每一个位占一个字符。对于负值，“–”占一个字符，若一个数的小数部分不是零时，小数点“.”占一个字符，反之，小数点不占字符。

从表 8.2 可以看出，各种类型数据占用的存储空间是不同的，取值范围较大的类型所需的存储空间也较大。

2．日期和时间类型

日期和时间类型用于处理时间数据，可以存储当日的时间或者出生日期这样的数据。表 8.3 列出了 MySQL 中可用的日期类型。

表 8.3　日期和时间类型

日期和时间类型	描述
DATE	日期。格式为‘YYYY-MM-DD’，日期范围‘1000-01-01’~‘9999-12-31’，占 3B 存储空间
DATETIME	日期时间。格式‘YYYY-MM-DD HH:MM:SS’。时间范围‘1000-01-01 00:00:00’~‘9999-12-31 23:59:59’，占 8B 存储空间
TIMESTAMP	时间戳。格式为‘YYYY-MM-DD HH:MM:SS’。如果要返回数字形式，则给 TIMESTAMP 列加+0。占 4B 存储空间
TIME	时间。格式为 HH:MM:SS，占 3B 存储空间
YEAR[(2|4)]	年份。2 表示年份为 2 位数字，值为 70~69，表示 1970—2069 内的年份。4 表示年份为 4 位数字，其值为 1901~2155。默认为 4 位的年份，占 1B 存储空间

3．字符串类型

字符串列用于存储任何类型的字符数据，如姓名、地址等。表 8.4 列出了 MySQL 可用的字符串类型。

（1）CHAR 和 VARCHAR 类型是类似的，但是它们被存储和检索的方式不同。

（2）一个 BLOB 列是一个能保存可变数量的数据的二进制对象。BLOB 的四种类型：TINYBLOB、BLOB、MEDIUMBLOB 和 LONGBLOB，它们仅在保存值的最大长度方面有所不同。

表 8.4　字符串类型

类型	描述
CHAR(M)	固定长度的字符串，M 为列长度，0≤M≤255。如果字符串长度<M，存储时在右边补齐空格，达到指定的长度。占 MB 存储空间
VARCHAR(M)	可变长度的字符串。一个可变长的字符串，其后缀空格在存储时被删除，0≤M≤255
TINYBLOB	最多 255（2^8–1）个字符的二进制对象，要求长度+1B 的存储空间
TINYTEXT	最多 255（2^8–1）个字符的文本，要求长度+1B 的存储空间
BLOB	二进制对象，最多 65 535(2^16 –1)个字符，要求长度+2B 的存储空间
TEXT	最多 65 535(2^16–1)个字符的文本，要求长度+2B 的存储空间
MEDIUMBLOB	最多 16 777 215(2^24–1)个字符的中等二进制对象，要求长度+3B 的存储
MEDIUMTEXT	最多 16 777 215(2^24–1)个字符的文本，要求长度+3B 的存储空间
LONGBLOB	最多 4 294 967 295 即 4GB (2^32–1)个字符的大二进制对象，要求长度+4B 的存储空间
LONGTEXT	最多 4 294 967 295 即 4GB (2^32–1)个字符的大文本，要求长度+4B 的存储空间
ENUM('value1','value2',…)	枚举。值为'value1'、'value2'等之一，或 NULL，或者错误时的空串"。一个枚举列最多 65 535 个不同的枚举值
SET('value1','value2',…)	集合。可以包含 0~64 个元素。值为‘value1’、‘value2’等之一

一个 TEXT 列是一个能保存可变数量的文本，TEXT 四种类型：TINYTEXT、TEXT、MEDIUMTEXT 和 LONGTEXT。它们对应于四种 BLOB 类型，并且有同样的最大长度和存储空间要求。BLOB 和 TEXT 类型之间的唯一差别是：对 BLOB 的值进行排序和比较时，要区分大小写，而 TEXT 值不用区分字母大小写。

（3）一个 ENUM 是一个字符对象，其值通常在创建表时定义的取值表中选择。它可以是空字符串（""）或者 NULL。

如果把一个无效值插入到 ENUM 列，或者 ENUM 列被赋值为 NULL，或者默认值为 NULL 而在插入记录时没有显式赋值，那么就得到值为 NULL 的 ENUM 列类型。

（4）一个 SET 列是可以有零或多个值的一个字符串对象，其中每一个值必须从创建表时定义的值列表中被选择。由多个集合成员组成的 SET 列由逗号分隔（“,”），因此 SET 成员值不能包含逗号本身。

例如，一个指定为 SET("one", "two") NOT NULL 的列可以存有下列任何一个值：

```
"", "one", "two", "one, two"
```

一个 SET 最多能有 64 个不同的成员。

8.4　MySQL 基本命令

前一节介绍了 MySQL 的基本语法，在本节以一个小型的学生管理系统为实例，介绍 MySQL 基本命令的具体应用。

学校每年都有新生入学、毕业生离校，还需要进行学生成绩录入。如何有效地管理这些学生的信息，帮助学校和老师掌握学生的情况，这就是学生信息管理系统需要完成的功能。通过学生信息管理系统实例的介绍，可以掌握 MySQL 的基本用法，为以后开发大型

的数据库系统打下良好的基础。

1．系统功能分析

本系统主要的功能是收集学生的个人信息，以便向教师提供每个学生在校的情况。系统的主要功能有：

- 学生个人信息输入，包括学号、姓名、性别、出生日期、专业、籍贯、高考分、联系电话、照片等。
- 学生课程表的输入，包括课程号、课程名、学分。
- 学生各科成绩的输入。
- 学生个人信息、成绩及所选课程的查询和修改。
- 用户权限的设定及修改。

2．数据库逻辑结构设计

假设本例的数据库文件名为 student_db。根据系统功能要求，需要设计三个数据表来存放学生的信息，这三个数据表分别是学生表 student、课程表 course 及学生选课成绩表 score，它们的记录内容分别如表 8.5、表 8.6 和表 8.7 所示。

表 8.5　学生表（student 表）

学号	姓名	性别	出生日期	专业	籍贯	高考分	联系电话	照片
0601	李大伟	男	1984.6.20	会计	河北	630.8	28718990	
0602	张莉莉	女	1985.3.28	软件工程	广西	627.0	37827862	
0603	李武	男	1983.10.16	会计	广州	619.5	37822891	
0604	王铭	男	1984.12.10	软件工程	广西	621.0	37828983	
0605	张晨	女	1983.1.5	工商管理	江西	630.0	23652121	
0606	黄勇	男	1984.9.20	软件工程	湖南	636.0	26778445	

表 8.6　课程表（course 表）

课程号	课程名	学分
1	数据库技术	4
2	高等数学	4
3	操作系统	3
4	数据结构	4
5	会计学原理	4
6	财务管理	3
7	管理学	3

表 8.7　成绩表（score 表）

学号	课程号	成绩
0601	5	85
0601	6	80
0602	2	90
0602	1	87
0603	5	84
0604	1	78
0605	7	74

8.4.1　创建和删除数据库

1．创建数据库

语法：

```
CREATE DATABASE 数据库名;
```

该语句用来创建一个新的数据库，当此数据库存在时，将返回一个错误。数据库的名字使用 MySQL 的命名规则。在 MySQL 中新建的数据库以一个目录形式存在硬盘中。执行该命令以后，MySQL 将在数据目录下创建一个新的目录，该目录名为新创建的数据库名，并且由于该数据库为空，目录下面没有任何文件。例如，创建学生管理数据库

student_db，命令如下：

```
mysql>CREATE DATABASE student_db;
```

2．打开数据库

在 MySQL 中，需要对某一个数据库的对象进行操作时，要先用 use 命令打开该数据库。use 命令格式为

语法：

```
use 数据库;
```

3．删除数据库

语法：

```
DROP DATABASE [IF EXISTS] 数据库名
```

该语句用来删一个数据库，并返回被删除文件的数目，这个数目通常是表数目的三倍，因为每个表对应 MYD、MYI、FRM 三个文件。如果想在执行该语句时不会发生错误，则可以在语句中加入关键词 IF EXISTS。它首先检查此数据库是否存在，如果存在才会执行删除命令，如果不存在则返回 0，不会发生没有找到数据库的错误。例如:

```
mysql>DROP DATABASE student_db;
```

8.4.2 创建和删除表

1．创建新表

语法：

```
CREATE  TABLE [IF NOT EXISTS] 表名 [(create_def,...)]
[table_options] [select_statement]
```

该语句用来在当前数据库中创建一个新表，该表的名字采用 MySQL 的命名规则。其中，主要的可选项有下面几个。

1）IF NOT EXISTS

如果指定的数据库不存在或该表名已经存在，则会返回一个错误，如果想避免在这种情况下返回错误，可以加入关键词 IF NOT EXISTS。

2）create_def 选项

create_def 部分定义表的结构，基本格式为：

```
列名 类型 [NOT NULL|NULL] [DEFAULT 值] [AUTO_INCREMENT] [Primary Key]
```

各子句的含义如下：

（1）NULL 或 NOT NULL 规定这个字段是否可以为 NULL 值，如果未指定，则各字段的默认值为 NULL。

（2）DEFAULT 子句指定字段的默认值。在插入新记录时，未指定该字段的值，一般插入默认值。

（3）AUTO_INCREMENT 用于整数类型，该字段的值从 1 开始，每次插入新记录时，

自动地以原来的最大值加 1 作为插入值。

（4）Primary Key 子句指定某一字段是该表的主键。

当一个表被创建时，MySQL 将在数据库对应的目录下为该表创建 student.frm（表结构文件）、student.myd（数据文件）、student.myi（索引文件）三个文件。在使用过程中用户要注意经常备份这些文件。

定义表时，指定的列类型可以为 MySQL 的所有数据类型。在定义的列中可以使用关键词 PRIMARY KEY 指定主键，主键是表中该列的各行数据是唯一的列。定义主键时要注意所定义的主键列必须同时定义为 NOT NULL，如果用户没有指定为 NOT NULL，那么 MySQL 自动地将该主键所在的列定义成 NOT NULL 属性。

一个表只能有一个主键。如果在表中没有一个主键，并且一些应用程序要求主键，MySQL 将返回第一个没有 NULL 值的 UNIQUE 列，作为表的主键。

【例 8.3】 在学生管理数据库 student_db 中创建学生表 student，命令如下：

```
USE student_db;
CREATE TABLE student (
    Sno varchar(4) NOT NULL default '',
    Sname varchar(10) default NULL,
    sex varchar(2) default NULL,
    birthdate date default '0000-00-00',
    specialty varchar(20) default NULL,
    native_place varchar(20) default NULL,
    gaokaofen decimal(5,1) default '0.0',
    phone varchar(15) default NULL,
    photo mediumblob,
    PRIMARY KEY  (Sno)
) ENGINE=MyISAM DEFAULT CHARSET=gb2312;
```

2．删除表

语法：

```
DROP TABLE [IF EXISTS] 表 1 [, 表 2,...]
```

此语句用来删除一个或多个数据库的表。表中的所有数据和表定义均被删除，并且如果没有备份表，则表是不能够恢复的，所以要小心使用这个命令。如果在执行此语句时不想因为表不存在而返回一个错误，那么就可以在语句中加上关键词 IF EXISTS。

例如，删除 student 表，命令为：

```
DROP TABLE student
```

如果只要删除表中的一个记录而不是整个表，则使用后面介绍的 DELETE 语句。

8.4.3 查看数据库名和表名

查看信息主要使用 SHOW 语句，它后面可以带不同的参数来查各种不同的信息，下面列举 show 命令的各种用法，并做简要讲解。

1．SHOW DATABASES [LIKE "字符串"]

列出所有数据库名或者符合通配符的数据库名。

2．SHOW TABLES [FROM 数据库名] [LIKE "字符串"]

列出当前数据库的所有表名或者符合通配符的表名。

3．SHOW COLUMNS FROM 表名[FROM 数据库名] [LIKE "字符串"]

列出指定数据库中指定表的所有列的信息。

4．SHOW INDEX FROM 表名 [FROM 数据库名]

列出数据库中指定表的索引信息。

5．SHOW STATUS

列出服务器的状态信息。

6．SHOW VARIABLES [LIKE "字符串"]

列出系统变量及其值。

7．SHOW [FULL] PROCESSLIST

列出正在运行的所有线程，如果不使用选项，那么每个查询只有头字符被显示出来。

8．SHOW TABLE STATUS [FROM 数据库名] [LIKE "通配符"]

列出数据库中每个表的详细信息。

9．SHOW GRANTS FOR user

列出对一个用户必须发出以重复授权的授权命令。

SHOW 命令提供关于数据库、数据表、字段列或服务器的信息。如果使用“LIKE "字符串"”子句，则"字符串"的内容可以是一个使用 SQL 的“%”和“_”通配符的字符串。

可以使用 student_db.student 作为 student FROM student_db 句法的另一种选择。下面两个语句是等价的：

```
mysql> SHOW INDEX FROM student FROM student_db;
mysql> SHOW INDEX FROM student_db.student;
```

SHOW DATABASES 列出在 MySQL 服务器主机上的数据库名，也可以执行 MySQL 客户端程序 mysqlshow，得到所有数据库名。

SHOW TABLE STATUS 运行类似 SHOW STATUS，但是提供每个表的更多信息。该命令返回下面的列：

列	含义
Name	表名
Type	表的类型（ISAM、MyISAM 或 HEAP）
Row_format	行存储格式（固定、动态或压缩）
Rows	行数量
Avg_row_length	平均行长度
Data_length	数据文件的长度
Max_data_length	数据文件的最大长度
Index_length	索引文件的长度
Data_free	已分配但未使用了字节数
Auto_increment	下一个 autoincrement（自动加 1）值
Create_time	表被创造的时间
Update_time	数据文件最后更新的时间

Check_time	最后对表运行一个检查的时间
Create_options	与 CREATE TABLE 一起使用的额外选项
Comment	表的注释

SHOW FIELDS 是 SHOW COLUMNS 的一个同义词，也可以用客户端程序 mysqlshow db1 tab1 或 mysqlshow -k db1 tab1 列出一张表的列或索引。

SHOW INDEX 命令返回索引信息，返回下面的列：

列	含义
Table	表名
Non_unique	0，如果索引不能包含重复
Key_name	索引名
Seq_in_index	索引中的列顺序号，从 1 开始
Column_name	列名
Collation	列的排序方式，在 MySQL 中，这可以有值 A（升序）或 NULL（不排序）
Cardinality	索引中唯一值的数量。这可通过运行 isamchk -a 更改
Sub_part	如果列只是部分被索引，返回索引字符的数量；如果整个键被索引，返回 NULL

SHOW VARIABLES 显示出一些 MySQL 系统变量的值，也能使用 mysqladmin variables 命令得到这个信息。如果默认值不合适，在启动 mysqld 时使用命令行选项来设置这些变量的大多数变量。SHOW PROCESSLIST 显示哪个线程正在运行，也可以使用 mysqladmin processlist 命令得到这个信息。如果有 process 权限，可以看见所有的线程。否则，仅能看见自己的线程。如果不使用 FULL 选项，那么每个查询只有头 100 字符被显示出来。

SHOW GRANTS FOR user 列出对一个用户必须发出以重复授权的授权命令。

【例 8.4】 SHOW GRANTS FOR user 语句的应用。

```
mysql> SHOW GRANTS FOR root@localhost;
+----------------------------------------------------------+
| Grants for root@localhost                                |
+----------------------------------------------------------+
| GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' WITH GRANT OPTION  |
+----------------------------------------------------------+
1 row in set (0.02 sec)
```

8.4.4 表的操作

1．插入记录

插入记录命令有三种基本格式：

格式 1：

```
INSERT [LOW_PRIORITY | DELAYED] [INTO] 表名[(字段1,字段2,…)]
VALUES (值1,值2,…);
```

- LOW_PRIORITY：可以使 INSERT 的执行被推迟到没有其他客户正在读取表时执行。在这种情况下，客户必须等到插入语句完成后才能往后操作。如果表被频繁使

用，它可能花很长时间。

- Delayed：可使要插入的记录不会马上写入数据库中，而是写入一个缓冲区（cache）中，等到要写入的数据达到一定的数目时，才一次性地写入到数据表中。

格式 1 形式的 INSERT 语句将明确指定的值作为一个新记录插入到表中。它可以一次插入多个记录。

如果未给出字段列表，则 VALUES 部分必须按表中字段顺序给出每个字段的值。如果只需要给新记录的部分字段赋值，那么 VALUES 部分的值个数、类型必须和字段列表的字段个数、类型、顺序要一致。例如：

```
mysql>INSERT INTO student (sname,sex) VALUES("张三","女");
```

格式 2：

```
INSERT [LOW_PRIORITY | DELAYED] [INTO] 表名[(字段1,字段2,…)]
SELECT 语句;
```

这种格式的 INSERT 语句将 SELECT 语句的查询结果作为新记录，插入到指定的表中。

格式 3：

```
INSERT [LOW_PRIORITY | DELAYED] [INTO] 表名[(字段1,字段2,…)]
SET 字段名 1=表达式 1,字段名 2=表达式 2, …;
```

格式 3 的 INSERT 语句采用直接指定字段的值这一方式，插入一个新记录。

2．删除记录

格式：

```
DELETE [LOW_PRIORITY] FROM 表名 [WHERE 条件] [LIMIT rows]
```

DELETE 命令从表中删除满足条件的行，并且返回被删除记录的行数。

如果发出一个没有 WHERE 子句的 DELETE 命令，那么所有行都被删除。MySQL 通过创建一个空表来完成，它比删除每行要快。在这种情况下，DELETE 返回零作为受影响记录的数目。

如果指定关键词 LOW_PRIORITY，DELETE 的执行被推迟到没有其他客户读取表后。

“LIMIT rows”选项告诉 MySQL 服务器将要删除的最大行数，这可以用来保证一个特定 DELETE 命令不会花太多的时间。可以简单地重复 DELETE 命令直到受影响的行数小于 LIMIT 值。

例如，删除刚才加入的 sname 名为“张三”的记录：

```
Delete from student Where sname="张三";
```

3．更新记录

格式：

```
UPDATE 表名 SET 字段 1=值 1,字段 2=值 2,... [WHERE 条件]
```

UPDATE 用新值更新现存表中行的列，SET 子句指出哪个列的值被修改为指定的值。如果给出 WHERE 子句，则更新满足条件的行中指定列的值，否则所有行被更新。

例如，下列语句设置 gaokaofen 列为它的当前值加 10：

```
mysql> UPDATE student SET gaokaofen=gaokaofen+10;
```

UPDATE 赋值是从左到右计算。例如，下列语句先将 gaokaofen 列的值扩大两倍，然后再加 1：

```
mysql> UPDATE student SET gaokaofen=gaokaofen*2, gaokaofen=gaokaofen+1;
```

灵活利用 UPDATE 语句对数据库进行操作，可以省去很多重复的计算工作。

4. 查询数据

格式：

```
SELECT [参数选项] 返回字段的名称
[FROM 表名]
[WHERE 条件]
[GROUP BY 字段名列表]
[HAVING 条件]
[ORDER BY 字段名列表]
[LIMIT [offset,] rows]]
```

各选项子句说明如下所述。

（1）参数选项：共有“All”、“Distinct”、“Distinctrow”三个选项可使用，此选项可用指定是否返回重复的记录，“All”表示返回全部重复的记录，“Distinct”及“Distinctrow”则表示删除重复的记录。

（2）返回字段的名称：指定要返回的字段名称，若是多个字段用逗号（,）分隔，若是“*”号则表示返回所有字段。在字段上可加上“As”指定字段别名，对字段名称较长者使用比较方便。

（3）FROM 表名：指出从哪个表中检索行。

（4）WHERE 条件：指定查询的记录应满足的条件，用户可以在条件中列出多个判断条件，在条件间可以使用逻辑运算符“AND”和“OR”。

例如，列出来自“广西”或者“河北”，且在 1984 年 1 月 1 日前出生的学生，可以使用下面的查询语句：

```
mysql> select sname,sex,birthdate,native_place from student
    -> where birthdate<"1984-01-01" and (native_place="广西"
    -> or native_place ="河北");
```

（5）ORDER BY 子句指定查询结果的排序规则。为了按某一字段的降序方式排列，在该字段后面加上 DESC 关键词。在默认情况下，按升序方式排列，这也可以用 ASC 关键词明确指定。例如：

```
mysql> select sname,birthdate from student order by birthdate desc;
```

（6）GROUP BY 子句指定分组规则，按照指定字段进行分组，将这些字段的值相同的所有记录分为一组。

（7）HAVING 子句与 GROUP BY 子句配合使用，它用于指定分组的记录是满足条件的记录。

例如，统计各专业的学生人数，命令为：

```
mysql> select specialty,count(*) from student group by specialty;
```

输出结果如下：

```
+-----------+----------+
| specialty | count(*) |
+-----------+----------+
| 工商管理   |        1 |
| 会计       |        2 |
| 软件工程   |        3 |
+-----------+----------+
3 rows in set (0.00 sec)
```

（8）LIMIT 子句用来限制 SELECT 语句返回的行数。LIMIT 取一个或两个数字参数，如果给定两个参数，第一个参数指定要返回的第一行的偏移量，第二个参数指定返回行的最大数目。初始行的偏移量是 0，而不是 1。

```
mysql> select * from student LIMIT 5,10;  # 查询第 6~第 15 行的记录
```

如果给定一个参数，它指出返回行的最大数目。

```
mysql> select * from student LIMIT 5;     # 查询前 5 个记录
```

换句话说，LIMIT n 等价于 LIMIT 0，n–1。

5. 修改表结构

修改表结构的命令是 ALTER TABLE 语句。它可以修改一个表的结构，例如增加或删除列、改变列的类型或重新命名列或表名，等等。其语法格式为：

格式：

```
ALTER  TABLE 表名 修改子句
```

下面给出几种常用的修改子句。

1）ADD [COLUMN] 列名 类型 [FIRST | AFTER column_name]

该子句在表的末尾增加一个新的字段。例如，在 student 表增加一个名为 tuanyuan 的新列，类型为 varchar(30)，命令为：

```
mysql> ALTER TABLE student ADD tuanyuan varchar(30);
```

2）CHANGE [COLUMN] 原列名 新列名 类型

该子句更改表中列的定义。例如，将刚才增加的 tuanyuan 列的类型改为 char(30)，命令为：

```
mysql> ALTER TABLE student CHANGE tuanyuan tuanyuan char(30);
```

3）DROP [COLUMN] 列名

该子句删除指定的列。例如，删除 tuanyuan 列，命令为：

```
mysql> ALTER TABLE student DROP tuanyuan;
```

4）RENAME [AS] 新表名

该子句重新命名表名。例如，将表名 student 改为 newstudent，命令为：

```
mysql> ALTER TABLE student RENAME newstudent;
```

ALTER TABLE 命令通过复制原来表的一个临时副本来工作。修改在副本上施行，然后原来的表被删除并且重新命名一个新的表。这样做使得所有的修改自动地转向到新表，没有任何失败的修改。当 ALTER TABLE 命令正在执行时，原来的表可以被其他客户读取。

上面仅列出 ALTER TABLE 命令的一些常用子句，其他子句可以参考 MySQL 帮助文档。

8.5 MySQL 权限

8.5.1 添加用户和设置权限

为 MySQL 数据库服务器增加一个新用户是最常用的操作之一，它可以为不同的用户设置不同的访问权限，并可以限制用户数据的访问范围。

可以用两种不同的方法来增加新用户：

- 使用 GRANT 语句；
- 通过 SQL 语句直接操作 MySQL 授权表。

比较好的方法是使用 GRANT 语句，因为 GRANT 语句更简明并且错误较少。GRANT 语句的格式如下：

```
GRANT priv_type [(column_list)] [, priv_type [(column_list)] ...]
    ON {tbl_name | * | *.* | db_name.*}
    TO user_name [IDENTIFIED BY 'password']
        [, user_name [IDENTIFIED BY 'password'] ...]
[WITH GRANT OPTION]
```

GRANT 和 REVOKE 命令允许系统管理员在四个权限级别上授权和撤回赋予 MySQL 用户的权限。

- 全局级别：全局权限作用于一个给定服务器上的所有数据库，这些权限存储在 mysql.user 表中。
- 数据库级别：数据库权限作用于一个给定数据库的所有表，这些权限存储在 mysql.db 和 mysql.host 表中。
- 表级别：表权限作用于一个给定表的所有列，这些权限存储在 mysql.tables_priv 表中。
- 列级别：列权限作用于在一个给定表的单个列，这些权限存储在 mysql.columns_priv 表中。

说明：

（1）对于 GRANT 和 REVOKE 语句，priv_type 参数指定权限类型，它可以是下列权限之一：

ALL PRIVILEGES	FILE	RELOAD
ALTER	INDEX	SELECT

CREATE	INSERT	SHUTDOWN
DELETE	PROCESS	UPDATE
DROP	REFERENCES	USAGE

其中，All 拥有所有权限，Alter 用于改变数据表与索引，Create 用于建立数据库和数据表，Delete 用于从数据表中删除数据，Drop 用于删除数据库或数据表，Insert 用于插入数据到数据表，Select 用于查询数据表，Update 用于修改数据表。

（2）tbl_name | * | *.* | db_name.*：表示权限所施加的层次，可用在所有数据库及数据表，也可以指定特定的数据库或数据字段。

（3）user_name：为特定的用户设置权限，它包括“用户主机”及“用户名”。

（4）Password：指定用户的密码。如果新增的用户没有指定密码，这代表该用户没有密码。

用户可以通过使用“ON *.*”语法设置全局权限，可以通过使用“ON db_name.*”语法设置数据库权限。如果用户指定“ON *”，并且此时用户已经有一个当前数据库，则用户将为该数据库设置权限。

注意：*如果指定“ON *”而没有一个当前数据库，则将影响全局权限。*

当安装完 MySQL 并正式开始使用时，便要根据不同用户的需要，为他们设置访问数据库的账号，例如，增加一个新用户：

```
mysql> grant all privileges on *.* to newuser @ "%"
    -> identified by "abc";
```

这个例子是增加一个新用户，newuser 是增加的用户账号，abc 是这个新用户的密码，如果不指定，则默认密码为空。

8.5.2 修改用户密码

刚安装完 MySQL 时，它只有一个用户：root。这个用户是所谓的超级用户，享有一切操作数据库的权限，既可以创建和删除数据库、创建和删除表格、数据检索和更新，还可以增加和删除用户等。该用户的初始密码为空，即没有密码，也可以通过如下 MySQL 命令来修改 root 用户的密码。

```
[root@myhost mysql]#.bin/mysqladmin -u root -password [newpassword]
```

“newpassword”项即是 root 用户的新密码。

还有一种修改用户密码的方法，就是在 MySQL 客户端程序窗口中直接修改 mysql 数据库中保存的密码。如修改上述 root 的密码，可以这样：

```
mysql>SET PASSWORD FOR root=PASSWORD("newpassword");
```

同上例一样，这里的“newpassword”也是指定新的 root 用户密码。

8.5.3 撤销用户权限

利用 REVOKE 命令撤销用户的权限。REVOKE 语句只删除权限而非用户，该用户的资料存于 user 数据表中，所以用户依旧可以连接到服务器上。要完全删除用户，必须在 user

数据表上使用 delete 语句，明确将用户记录删除。REVOKE 命令的格式为：

```
REVOKE priv_type [(column_list)] [, priv_type [(column_list)] ...]
   ON {tbl_name | * | *.* | db_name.*}
   FROM user_name [, user_name ...]
```

REVOKE 命令中各部分的含义和 GRANT 命令相同。例如：

```
mysql> revoke all on student.* from linxiaojun@localhost;
Query ok, 0 rows affected (0.00 sec)
```

REVOKE 语句只是撤销用户的某种权限。如果希望彻底地删除该用户，应当直接对 user 表进行删除记录操作，例如：

```
mysql> delete from user where user=linxiaojun and host=localhost;
   Query ok, 1 rows affected (0.00 sec)
```

【例 8.5】 完全删除用户的例子。

```
mysql> grant all on * from  wu@aasir.com;
Query ok, 0 rows affected (0.00 sec)
mysql> delete from user where user="wu" and host="aasir.com";
Query ok, 1 rows affected (0.00 sec)
mysql> flush privileges;
Query ok, 0 rows affected (0.00 sec)
```

8.5.4 备份和恢复数据库

数据库常常可能发生数据丢失或者损坏的情况，比如当用户更新完数据表后，实际被更新的数据还没有被刷新到磁盘中，此时系统崩溃，则这部分数据很可能没有被真正的保存，数据可能出现不一致的现象。因此，必须对数据库进行周期性地备份。可以说，备份是任何数据库系统中至关重要的组成部分。在 MySQL 数据库系统中，提供以下几种常用的备份和恢复数据库的方法。

1．用 BACKUP 命令备份 MyISAM 表

一种最容易的备份方法是用 BACKUP 命令，这只有在使用 MyISAM 表时才起作用。其语句结构如下：

```
BACKUP TABLE 表名 TO '存储路径'
```

存储路径是存放备份数据的目录路径。该命令将生成以 FRM（定义）和 MYD（数据）为扩展名的文件，但不产生.MYI（索引）文件。可以在数据库恢复后重建索引。

在处理文件时，要注意文件权限。备份时，如果没有全部文件和目录的恰当权限，则 MySQL 多数不会给出友好的错误信息以示警告。

【例 8.6】 在 UNIX 环境下使用 BACKUP 命令。以超级用户 root 进入根目录，执行以下命令，将 student_db 数据库的 student 表备份到/db_backup 目录下。

```
# cd /
# mkdir db_backup
```

```
# chown mysql db_backup/
# mysql student_db -u root -p123456
mysql> backup table student to '/db_backup';
```


注意：必须执行上面的 chown 命令，将 db_backup 目录的所有者设置为 mysql 用户，才能有权限备份表文件，否则将显示错误信息。

也可以在一个 BACKUP 命令中备份多个表，如：

```
mysql> backup table student,course to '/db_backup';
```

该命令将 student 表和 course 表备份到/db_backup 目录。

【例 8.7】 在 Windows 环境下用 BACKUP 命令，命令如下。

```
D:\>MKDIR db_backup
D:\mysql\bin>mysql student_db -u root -p123456
mysql> BACKUP TABLE course,student TO 'd:\\db_backup';
```

注意：在 Windows 环境下，表示路径的盘号和目录名之间以及目录名之间必须用两个反斜线符（\\）连接。

2．用 RESTORE 恢复 MyISAM 表

RESTORE 命令用来恢复以前使用 BACKUP 创建的 MyISAM 表。其语法格式如下：

```
RESTORE TABLE 表名 FROM '备份文件的存储路径'
```

【例 8.8】 在 UNIX 环境下恢复例 8.6 的 MyISAM 表，命令如下：

```
mysql> DROP TABLE student;
mysql> RESTORE TABLE student FROM '/db_backup';
```

注意：如果数据库中已经存在备份的表，则试图恢复表时将显示错误信息。

3．通过直接复制文件的方法备份 MyISAM 表

MyISAM 表以文件的形式存放在以数据库名命名的目录中（.frm 代表定义文件，.myd 代表数据文件，.myi 代表索引文件），所以备份数据表的一个简便方法是复制这些文件。这种方法与 BACKUP 命令不同，直接复制文件不会自动给这些表加锁。因此为了得到相关联的需要锁住这些表。另一种选择是在服务器关闭时直接复制文件。

例如，假设 MySQL 的 student_db 数据库的表文件存放在 LINUX 系统的/usr/local/mysqldata/student_db 目录下。以下命令将 student_db 数据库的所有表复制到/db_backup 目录下：

```
# cd /usr/local/mysqldata/student_db
# cp * /db_backup
```

注意：利用这种方法恢复数据表时，需要将数据库目录下的文件的所有者设置为 mysql 用户，这样才能在 MySQL 中访问这些恢复的表数据。

4．用 mysqldump 备份

前面两种备份表文件的方法只能在使用 MyISAM 表的时候起作用。对于 InnoDB 表来说，由于其不能以文件的形式保存，因此前面的方法无法使用。对于这种情况，可以采用

Mysqldump 程序，它用来生成备份表的 SQL 语句。

例如，在 Linux 超级用户的环境下，执行以下命令，生成的 SQL 语句存放到 student.sql 文件。

```
#mysqldump student_db student>/db_backup/student.sql;
```

或者在 Windows 下，执行以下命令：

```
D:\MySQL\bin>mysqldump student_db student >d:\db_backup\student.sql
```

需要时，还需指出表的路径、用户名和密码。

5. 恢复用 mysqdump 备份的数据库

为了恢复 UNIX 环境的表文件，可以执行以下命令：

```
mysql> DROP TABLE student;
mysql-> exit
#mysql student </db_backup/student.sql
```

至此，student 表已经被恢复了。

6. 用 SELECT INTO 语句备份数据

利用 SELECT INTO 语句也可以将查询的结果备份到服务器的某一个目录中。它与 LOAD DATA INTO 语句的作用相反。其语法格式如下：

```
SELECT ... INTO OUTFILE '路径/文件名'
```

例如，在 UNIX 下，为了备份 course 表，可以 SELECT 语句来实现，如下：

```
mysql> SELECT * FROM student INTO OUTFILE "/db_student/student.dat"
```

7. 用 LOAD DATA 命令恢复表

为了恢复由 SELECT INTO 命令建立的表，要使用 LOAD DATA 语句，也可以用 LOAD DATA 添加由其他方法建立的数据，如某个应用程序或电子数据表。这是最快的添加数据的方法，特别是对于大量数据。其基本语法如下：

```
LOAD DATA [LOCAL] INFILE '文件名' INTO TABLE 表名
```

例如，在 UNIX 下，先删除 student 表的数据，然后恢复该表的内容，命令如下：

```
mysql> TRUNCATE student;
mysql> LOAD DATA INFILE "/db_backups/" INTO TABLE  student;
```

8.6 基于浏览器的 MySQL 数据库管理工具——phpMyAdmin 工具

在过去的几年中，Linux、Apache、MySQL 和 PHP 已经成为 Web 数据库应用程序的事实标准。普遍认为，MySQL 是一套易于安装、快速且稳定可靠的关系数据库系统，但是，它的标准发行版本却只能使用命令行管理工具。直到现在，还没有一个高效、友好、

独立于系统平台的用户接口。基于浏览器的 MySQL 用户接口为数不多，而 phpMyAdmin 正是其中之一。如果打算提高工作效率，安装和使用 phpMyAdmin 是很有必要的。

8.6.1 phpMyAdmin 的特性

使用 phpMyAdmin 和网络浏览器，Web 开发人员可以很方便地完成大部分数据库管理员需要完成的任务，而不用关心 SQL 语句的语法。除了一些基本的功能，如创建数据库或表、编辑表的内容等，phpMyAdmin 还有另一些很有用的特性，它可以建立表或数据库，使用逗号分隔的方式导入和导出文本文件，并且支持十种以上的语言。

下面是它的主要功能列表：

（1）创建和删除数据库；

（2）创建、复制、删除、修改表；

（3）删除、编辑、添加字段；

（4）执行任何 SQL 语句，包括批查询；

（5）管理字段中的键值；

（6）将文本文件输入到表。

8.6.2 phpMyAdmin 实例讲解

在启动 phpMyAdmin 之前，一定要确认 Apache 服务器及 MySQL 数据库的进程正常运行。因为 phpMyAdmin 的文件都是 PHP 格式，需要 Apache 的支持，而 MySQL 也是必须的。启动时，打开 Internet Explorer 浏览器，在地址栏中输入 http://localhost/phpmyadmin/index.php。如果配置没有问题，将出现图 8.2 所示的初始界面。



图 8.2 phpMyAdmin 的初始界面

下面将讲解如何利用 phpMyAdmin 创建、删除数据库，创建、复制、删除、修改表、编辑、添加字段，执行任何 SQL 语句等操作。

1．创建数据库、表并添加记录

数据库是存储信息的仓库，以一种简单规则的方式进行组织。数据库中的数据集组织

为表，每个表由行和列组成。表中每行为一个记录，记录可包含几段信息：表中每一列对应这些信息中的一个字段。这就是最简单的数据库、表、记录三者之间的关系。下面从最外围的数据库说起。

在 phpMyAdmin 里，创建数据库是很方便的。在图 8.2 所示的初始界面中可见“创建一个新的数据库”的文本框，在文本框输入希望建立的数据库名称（比如 studentnew），单击“创建”按钮，可见图 8.3 所示的界面。

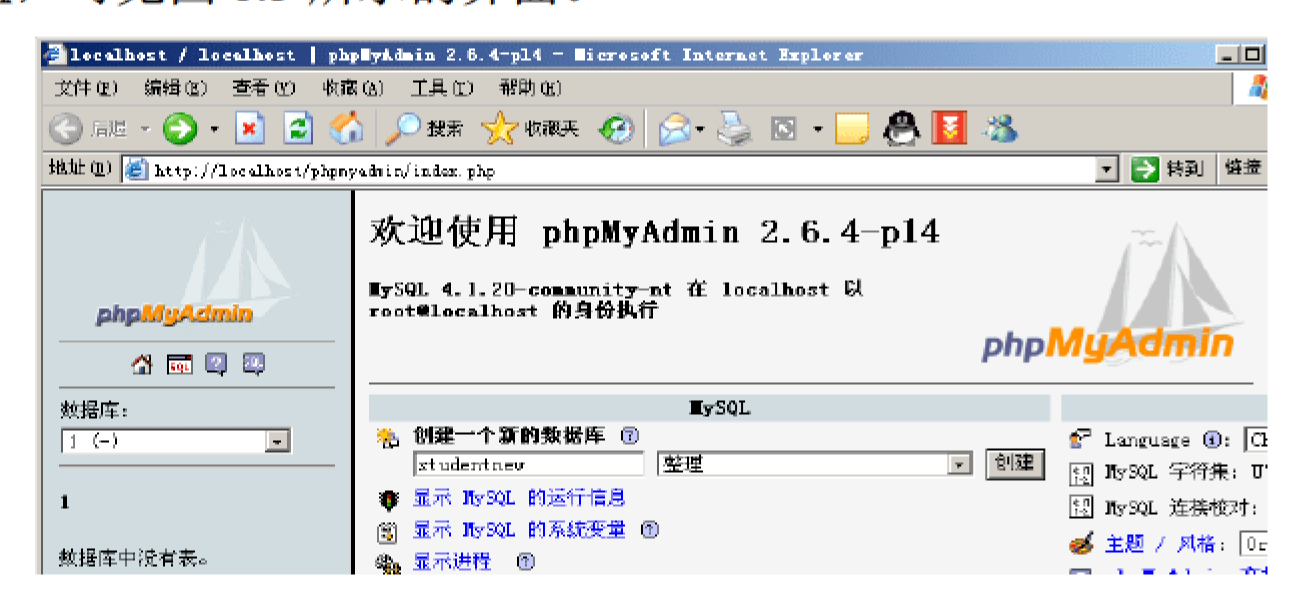


图 8.3　创建数据库的界面

片刻后，就在 MySQL 中建立了新的 studentnew 数据库。同时，左边的数据库列表中也添加了“studentnew”数据库的链接。

可以到默认目录中看一下，是不是多了一个名为“studentnew”的目录？只不过现在这个目录还是空的，因为这个数据库中还没有任何数据。现在，再向这个数据库中加入几个表，让它成为真正意义上的数据库。

在图 8.4 所示的界面中，单击左边“studentnew”的链接，在右边就出现“studentnew”数据库的描述页面。在这页的最下面，有“在数据库 studentnew 中创建一个新表”的文字提示信息。在“名字”右边的文本框中，输入希望建立的数据表的名称；在“字段数”右边的文本框中，输入希望建立的字段数。

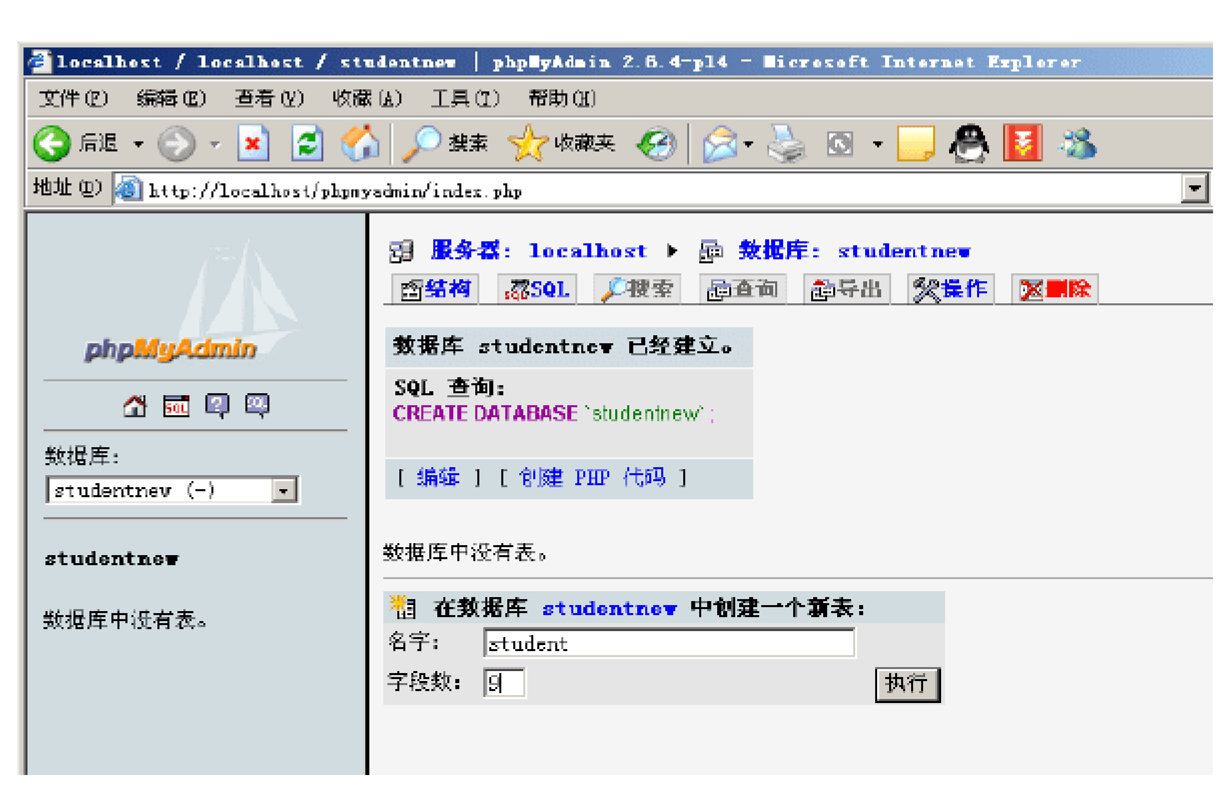


图 8.4　创建数据表的界面

单击“执行”按钮，就创建了一个名为“student”的表。在接下来打开的图 8.5 中，要求输入表结构的页面，包括 Sno、Sname、sex、birthdate、specialty、native_place、gaokaofen、

phone、photo 字段。在“字段”的文字输入框中输入“sno”，在“类型 ”中选择数据类型“VARCHAR”，选择表结构的相关值或输入需要的值。

图 8.5 表结构输入的界面

下面解释其中各项的含义。

字段：字段名，也就是以后每列的名称。

类型：字段数据类型，MySQL 支持 INT、FLOAT、DOUBLE、CHAR、VARCHAR、TEXT、BLOB、DATE、TIME、DATETIME、TIMESTAMP、YEAR、SET、ENUM 等多种数据类型。

长度/值：字段数据的长度，需要注意的是 TEXT、BLOB、DATE、TIME、DATETIME 类型的数据是不需要指定长度的。

属性：字段数据的属性，指定 INT 类型是有符号 / 无符号整数。

Null：字段数据是否可以为空值。

默认：字段数据的默认值。

额外：设定字段数据是否为自动增加（auto_increment），建议对“ID”这样的连续排列的整数字段使用该属性。

创建表结构完成，单击“保存”按钮，然后就会出现图 8.6 所示的界面，这个表就建好了。



字段	类型	整理	属性	Null	默认	额外
Sno	varchar(4)	gb2312_chinese_ci		否		
Sname	varchar(10)	gb2312_chinese_ci		是	NULL	
sex	varchar(2)	gb2312_chinese_ci		是	NULL	
birthdate	date			是	0000-00-00	
specialty	varchar(20)	gb2312_chinese_ci		是	NULL	
native_place	varchar(20)	gb2312_chinese_ci		是	NULL	
gaokaofen	decimal(5,1)			是	0.0	
phone	varchar(15)	gb2312_chinese_ci		是	NULL	
photo	mediumblob		BINARY	是	NULL	

图 8.6 表结构创建完成后的界面

在表操作部分，可以通过“浏览”命令浏览整个表，利用“SQL”命令进行 SQL 语句的自行输入，可通过“插入”命令插入记录，同时还可以插入新的字段，这些几乎包括了所有 SQL 语言的命令，在这里可以方便地修改表的各项属性及进行各种操作。

2．记录、表、数据库的修改及删除

当需要修改数据或删除多余记录时，也可以非常方便地使用 phpMyAdmin 的界面进行操作。直接单击表中“浏览”命令，此时会出现如图 8.7 所示的页面。

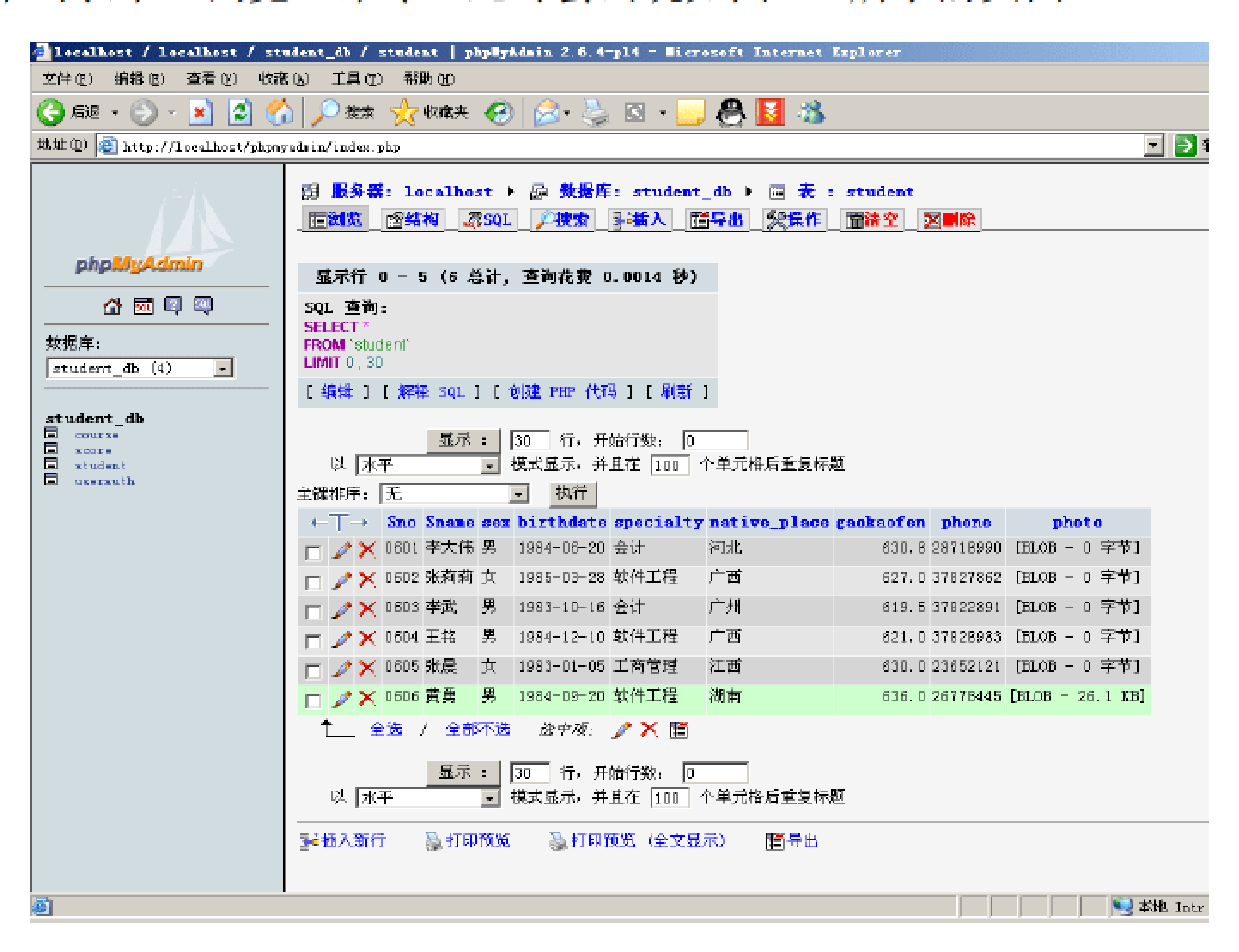


图 8.7　数据表浏览界面

如果想修改数据，在浏览页面先选中该记录左边的复选框，再单击“更改”按钮，然后在打开的图 8.8 中将所需信息进行修改，单击“执行”即可。

有时候要对数据库进行清理操作，把整个数据库的内容全部删除，使 MySQL 数据库中不包含任何的数据，那应该怎么做呢？这就是数据库的删除操作。只要选择“删除”命令，这时出现如图 8.9 所示的界面。



图 8.8　修改数据的界面



图 8.9　删除数据库的提示界面

接着，选择“确定”按钮，就可以了。

除了上述所列的功能以外，phpMyAdmin 还可以进行表与数据库的浏览、清空等的基本操作及网络安全的设置。

实　验　8

使用 MySQL 和 PHP 建立一个简单的用户认证。

习　题　8

1．请叙述数据库各层次的结构是什么？

2．请叙述 MySQL 数据库有哪些字段数据类型？并将每种数据类型列举两个较常用的形式，说明名称、存储大小及范围。

3．在 MySQL 数据库中，如果要新增一位名为“John”的用户，并使其对 localhost 上的 db 授权表拥有全部的存取权限，及使用密码“abc”进出数据库，请问该如何下达命令？

4．试叙述 Create Table 及 Alter Table 命令的功能及差异。

5．请写出下列 MySQL 命令片段的执行结果：

```
mysql>select trim(leading "a" from "aaabacaa");
```

第 9 章　PHP 5 的 MySQL 数据库编程

本章导读

Web 应用系统与其他应用系统一样，通常也需要数据库的支持。对于 PHP 来说，最常用的数据库就是 MySQL。实际上，“Apache+PHP+MySQL”早已成为 Web 应用开发中的首选模式之一。本章首先介绍 MySQL 数据库编程的基本步骤，然后再分别介绍在 PHP 5 中使用 mysql 函数库与 mysqli 函数库进行数据库编程的有关技术。

9.1　MySQL 数据库编程的基本步骤

在 PHP 5 程序中，为了实现对 MySQL 数据库的各种操作，既可以使用原有的 mysql 函数库，也可以使用新增的 mysqli 函数库（该函数库只适用于 MySQL 4.0 及以上版本）。但不管使用哪个函数库，其编程的基本步骤都是一样的。

在 PHP 中，MySQL 数据库编程的基本步骤如下：

（1）建立与 MySQL 数据库服务器的连接。

（2）选择要对其进行操作的数据库。

（3）执行相应的数据库操作，包括记录的检索、增加、修改、删除等。

（4）关闭与 MySQL 数据库服务器的连接。

对于以上所述的各个编程步骤，均可通过调用相应的函数加以实现。因此，要在 PHP 中实现对 MySQL 数据库的各种操作，就必须熟练掌握 MySQL 或 mysqli 函数库中的有关函数及其使用方法。

9.2　使用 MySQL 函数库进行数据库编程

在 PHP 中为了使用 mysql 函数库访问 MySQL 数据库，需要在 PHP 的配置文件 php.ini 中将“;extension=php_mysql.dll”修改为“extension=php_mysql.dll”（即删掉该选项前面的注释符号“;”），然后再重新启动 Web 服务器（如 Apache 等）。

9.2.1　建立与数据库服务器的连接

在 PHP 中，可使用 mysql_connect()函数，建立与 MySQL 数据库服务器的连接。其语法格式为：

```
mysql_connect([server[,username[,password]]])
```

参数：server（string 型）为 MySQL 数据库服务器的名称，其默认值为 localhost；username（string 型）为用户名，其默认值为超级用户 root；password（string 型）为密码，其默认值为空字符串。

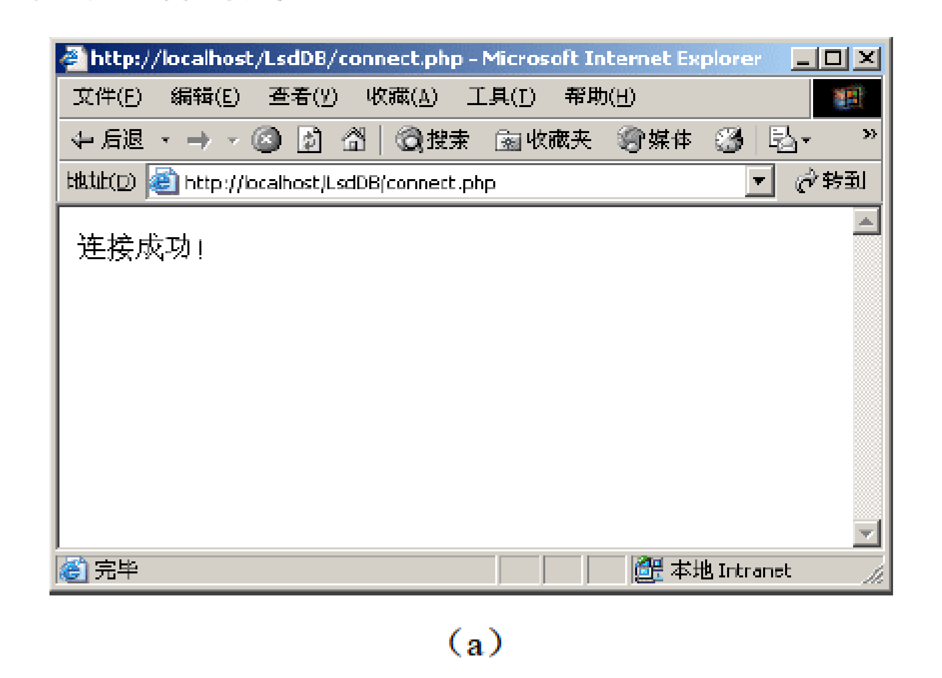
返回值：resource 型。若执行成功，则返回一个连接标识号（link_identifier）；否则，返回 FALSE。

在 PHP 程序中，通常要将 mysql_connect()函数返回的连接标识号保存在某个变量中，以备后用。实际上，在后续有关操作所调用的函数中，一般都要指定相应的连接标识号。

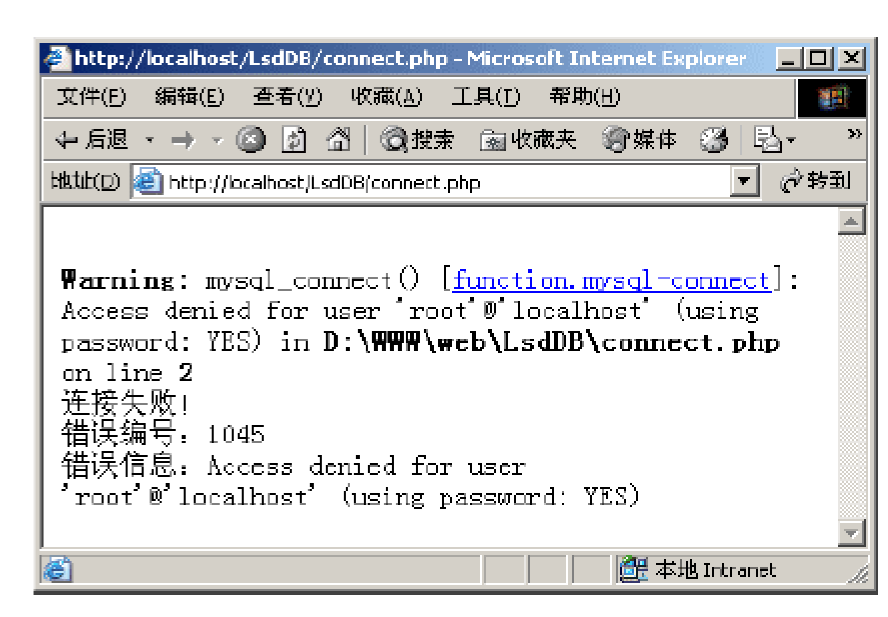
【例 9.1】 数据库服务器的连接示例（connect.php）。

```
<?php
  $link=mysql_connect("localhost","root","123456");
  if (!$link)  //若连接失败，则显示相应信息并终止程序运行
  {
      echo "连接失败！<BR>";
      echo "错误编号：".mysql_errno()."<BR>";
      echo "错误信息：".mysql_error()."<BR>";
      die();  //终止程序运行
  }
  echo "连接成功！<BR>";
?>
```

在该示例中，以超级用户 root（在本章中假定其密码为“123456”）连接本地主机中的 MySQL 数据库服务器。若连接成功，则显示“连接成功！”的信息，如图 9.1（a）所示。若连接失败，则显示相应的错误信息并终止程序的运行，如图 9.1（b）所示即为密码不正确时的运行结果。



（a）



（b）

图 9.1　程序 connect.php 的运行结果

在 PHP 中，一切非 0 值均为 TRUE，而 0 值则为 FALSE。mysql_connect()函数执行成功后所返回的连接标识号其实就是一个非 0 值，相当于 TRUE。因此，若要判断是否已成功建立与 MySQL 数据库服务器的连接，只需判断 mysql_connect()函数的返回值的真假即可。如果连接失败，那么可进一步调用 mysql_errno()与 mysql_error()函数获取相应的错误编号与错误信息。

mysql_errno()与 mysql_error()函数的功能分别为获取上一个 MySQL 函数（但不包括这

两个函数）执行后的错误编号与错误信息，若未出错则分别返回零（0）与空字符串（""）。因此，使用 mysql_errno()或 mysql_error()函数也可判断 mysql_connect()函数或其他 MySQL 函数的执行情况（成功或失败）。

建立连接是执行其他数据库操作的前提条件，因此在执行 mysql_connect()函数后，应立即进行相应的判断，以确定连接是否已成功建立。

9.2.2 选择数据库

一个数据库服务器往往会包含有为数众多的数据库，因此在执行具体的数据库操作之前，应首先选中相应的数据库。在 PHP 中，要选中某个 MySQL 数据库，可使用 mysql_select_db()函数。其语法格式为：

```
mysql_select_db(database_name[,link_identifier])
```

参数：database_name（string 型）为数据库名称；link_identifier（resource 型）为连接标识号，用于指定相应的与 MySQL 数据库服务器的连接。若未指定 link_identifier，则使用上一个打开的连接。若无打开的连接，则不带参数调用 mysql_connect()函数来尝试打开一个连接并使用之。

返回值：bool 型。若执行成功，则返回 TRUE；否则，返回 FALSE。

【例 9.2】 数据库的选择示例（selectdb.php）。

```
<?php
  $link=mysql_connect("localhost","root","123456");
  if (mysql_errno())    {
      echo "数据库服务器连接失败！<BR>";
      die();
  }
  mysql_select_db("mysql",$link);  //选择系统数据库 mysql
  if (mysql_errno())
  {
      echo "数据库选择失败！<BR>";
      die();
  }
  echo "数据库选择成功！<BR>";
?>
```

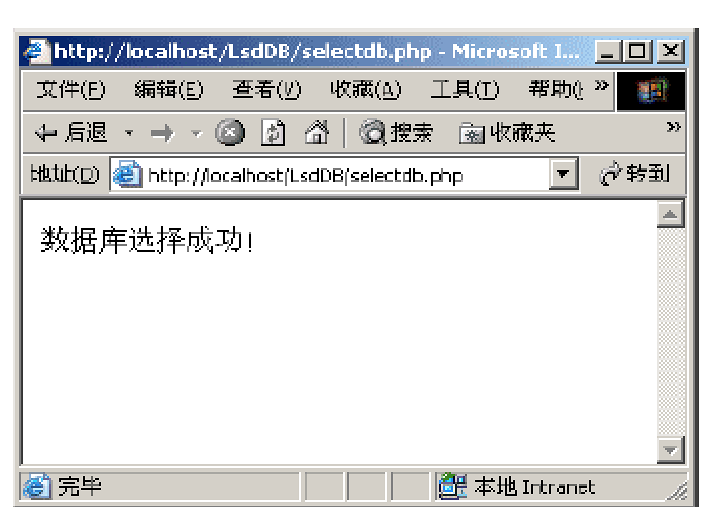


图 9.2 程序 selectdb.php 的运行结果

该示例的运行结果如图 9.2 所示。

必要时，也可在程序中直接创建所需要的数据库，或删除不再需要的数据库。为了创建数据库，可使用 mysql_create_db()函数。为了删除数据库，可使用 mysql_drop_db()函数。其语法格式为：

```
mysql_create_db(database name[,link_identifier])
mysql_drop_db(database_name[,link_identifier])
```

参数与返回值：与 mysql_select_db()函数相同。

【例 9.3】 数据库的创建与删除示例（createdropdb.php）。

```
<?php
$link=mysql_connect("localhost","root","123456")
    or die("数据库服务器连接失败！<BR>");
echo "数据库服务器连接成功！<BR>";
//创建数据库 test_db
mysql_create_db("test_db",$link) or die("数据库创建失败！<BR>");
echo "数据库创建成功！<BR>";
//选择数据库 test_db
mysql_select_db("test_db",$link) or die("数据库选择失败！<BR>");
echo "数据库选择成功！<BR>";
//删除数据库 test_db
mysql_drop_db("test_db",$link) or die("数据库删除失败！<BR>");
echo "数据库删除成功！<BR>";
?>
```

该示例的运行结果如图 9.3 所示。

应该注意的是，若 MySQL 函数库是基于 MySQL 4.x 客户端库建立的，则不允许使用 mysql_create_db()与 mysql_drop_db()函数。实际上，数据库的创建与删除也可以通过执行相应的 SQL 语句来实现。

图 9.3 程序 createdropdb.php 的运行结果

9.2.3 执行数据库操作

选中某个数据库后，即可对该数据库执行各种具体的操作，如记录的检索、增加、修改与删除以及表的创建与删除等。对数据库的各种操作，都是通过提交并执行相应的 SQL 语句来实现的。在 PHP 中，使用 mysql_query ()函数提交并执行 SQL 语句。其语法格式为：

```
mysql_query(query_statement[, link_identifier])
```

参数：query_statement（string 型）为相应的 SQL 语句；link_identifier（resource 型）为连接标识号，用于指定相应的与 MySQL 数据库服务器的连接。若未指定 link_identifier，则使用上一个打开的连接。若无打开的连接，则不带参数调用 mysql_connect()函数来尝试打开一个连接并使用之。

返回值：resource 型。对于 SELECT、SHOW、EXPLAIN 或 DESCRIBE 语句，若执行成功，则返回相应的结果标识符，否则返回 FALSE；对于 INSERT、DELETE、UPDATE 、REPLACE、CREATE TABLE、DROP TABLE 或其他非检索语句，若执行成功，则返回 TRUE，否则返回 FALSE。

下面，将通过一些具体的实例来说明如何通过编程的方式实现有关的数据库操作。

1．数据库的创建与删除

为创建数据库，只需执行相应的 CREATE DATABASE 语句即可。反之，为删除数据库，只需执行相应的 DROP DATABASE 语句即可。

【例 9.4】 数据库的创建与删除示例（query_db.php）。

```
<?php
$link=mysql_connect("localhost","root","123456")
    or die("数据库服务器连接失败! <BR>");
echo "数据库服务器连接成功! <BR>";
mysql_query("CREATE DATABASE test_db",$link)
    or die("数据库 test_db 创建失败! <BR>");
echo "数据库 test_db 创建成功! <BR>";
mysql_select_db("test_db",$link)
    or die("数据库 test_db 选择失败! <BR>");
echo "数据库 test_db 选择成功! <BR>";
mysql_query("DROP DATABASE test_db",$link)
    or die("数据库 test_db 删除失败! <BR>");
echo "数据库 test_db 删除成功! <BR>";
?>
```

该示例的运行结果如图 9.4 所示。

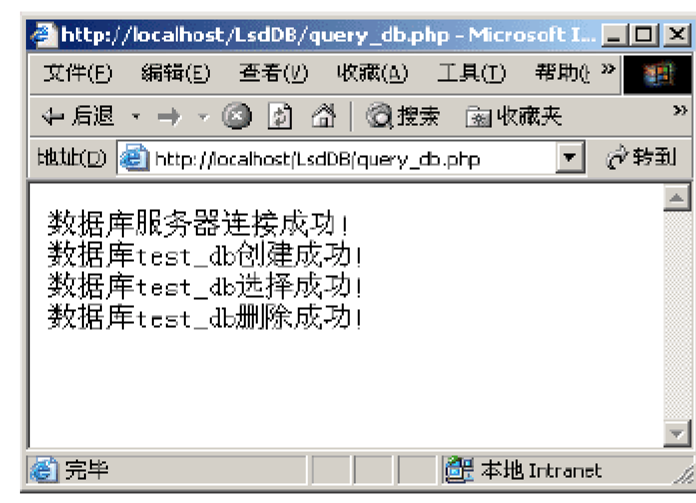


图 9.4　程序 query_db.php 的运行结果

2．表的创建、修改与删除

为创建表，只需执行相应的 CREATE TABLE 语句即可。为修改表，只需执行相应的 ALTER TABLE 语句即可。为删除表，只需执行相应的 DROP TABLE 语句即可。

【例 9.5】 表的创建示例（query_table_create.php）。

```
<?php
$link=mysql_connect("localhost","root","123456")
    or die("数据库服务器连接失败! <BR>");
mysql_select_db("mysql",$link) or die("数据库选择失败! <BR>");
//创建表 test_table
$sql="CREATE TABLE test_table(xh CHAR(9) NOT NULL,";
$sql=$sql."xm VARCHAR(20) NOT NULL,xb CHAR(2) NOT NULL,";
$sql=$sql."PRIMARY KEY(xh))";
mysql_query($sql,$link) or die("表 test_table 创建失败! <BR>");
echo "表 test_table 创建成功! <BR>";
?>
```

【例 9.6】 表的修改示例（query_table_alter.php）。

```
<?php
$link=mysql_connect("localhost","root","123456")
    or die("数据库服务器连接失败! <BR>");
mysql_select_db("mysql ",$link) or die("数据库选择失败! <BR>");
//在表 test_table 中添加字段 bj
$sql="ALTER TABLE test_table ADD bj CHAR(3) NOT NULL";
mysql_query($sql,$link)  or die("字段添加失败! <BR>");
//在表 test_table 中修改字段 bj
```

```
$sql="ALTER TABLE test_table CHANGE bj bj CHAR(5) NOT NULL";
mysql_query($sql,$link) or die("字段修改失败！<BR>");
//在表 test_table 中删除字段 bj
$sql="ALTER TABLE test_table DROP bj";
mysql_query($sql,$link) or die("字段删除失败！<BR>");
echo "表 test_table 修改成功！<BR>";
?>
```

【例 9.7】 表的删除示例（query_table_alter.php）。

```
<?php
$link=mysql_connect("localhost","root","123456")
    or die("数据库服务器连接失败！<BR>");
mysql_select_db("mysql ",$link) or die("数据库选择失败！<BR>");
//删除表 test_table
$sql="DROP TABLE test_table";
mysql_query($sql,$link) or die("表 test_table 删除失败！<BR>");
echo "表 test_table 删除成功！<BR>";
?>
```

3．示例数据库与表的创建

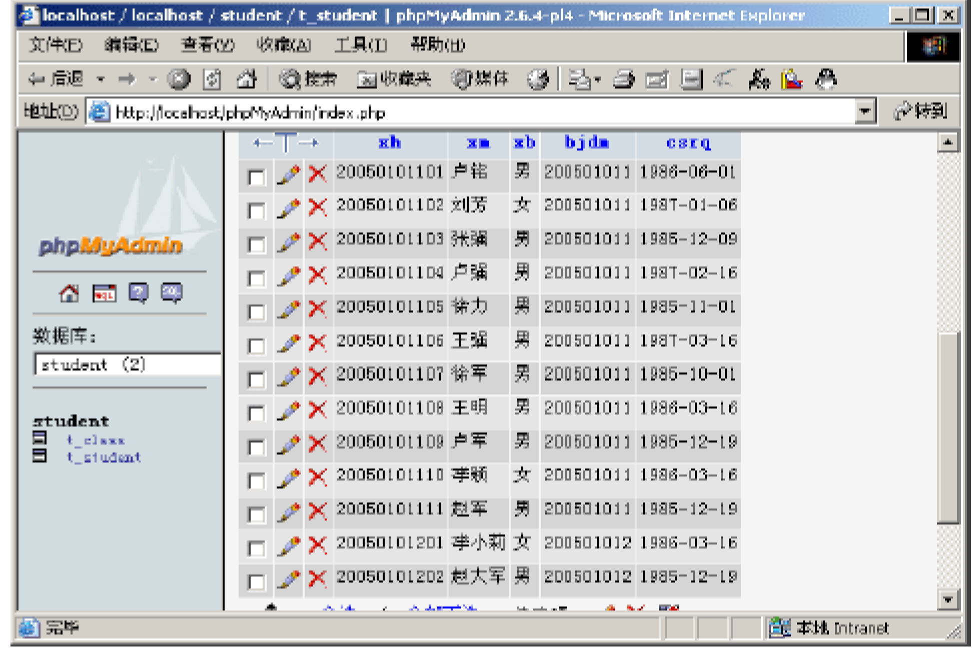
为便于说明数据库有关操作的编程技术，在此首先创建一个示例数据库 student，并在其中创建两个表——学生表 t_student 与班级表 t_class，各表的结构如表 9.1 和表 9.2 所示，然后再使用 MySQL 的管理程序 phpMyAdmin 输入相应的数据，如图 9.5 所示。创建示例数据库与表的程序代码如例 9.8 所示。

表 9.1 班级表 t_class 的结构

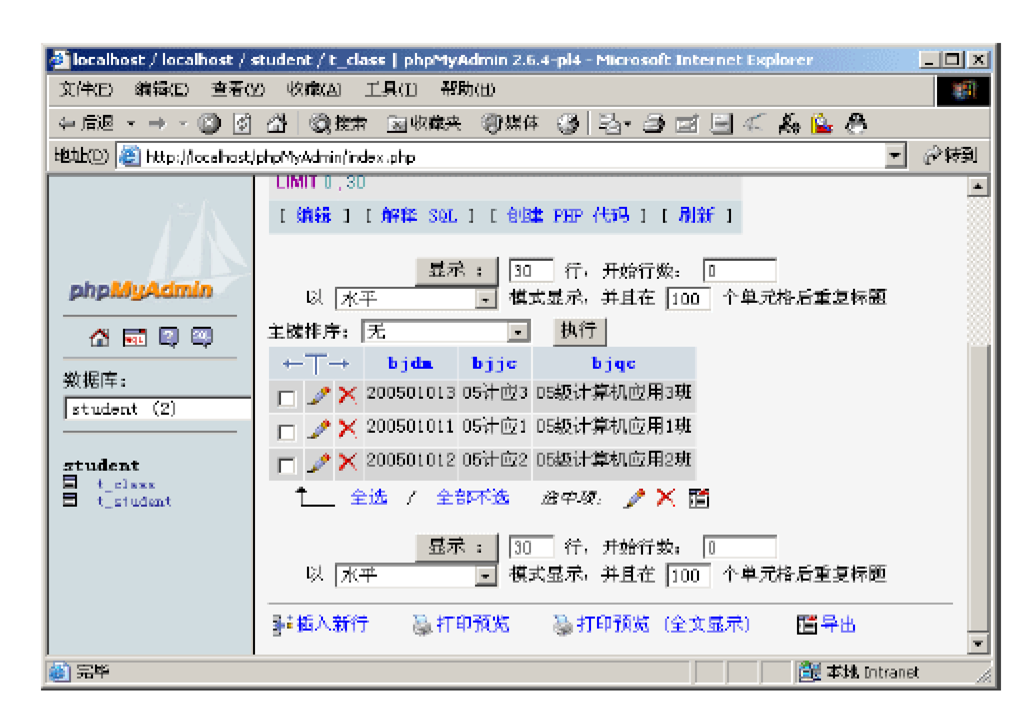
字段名	字段类型	字段说明
bjdm	char(9)	班级代码（主键）
bjjc	varchar(15)	班级简称
bjqc	varchar(30)	班级全称

表 9.2 学生表 t_student 的结构

字段名	字段类型	字段说明	字段名	字段类型	字段说明
xh	char(11)	学号（主键）	bjdm	char(9)	班级代码
xm	char(12)	姓名	csrq	date	出生日期
xb	char(2)	性别			



（a）



（b）

图 9.5 学生表与班级表的数据

【例 9.8】 示例数据库与表的创建（query_db_table.php）。

```
<?php
$link=mysql_connect("localhost","root","123456")
    or die("数据库服务器连接失败! <BR>");
//创建数据库 student
mysql_query("CREATE DATABASE student",$link)
    or die("数据库 student 创建失败! <BR>");
echo "数据库 student 创建成功! <BR>";
mysql_select_db("student",$link)
    or die("数据库 student 选择失败! <BR>");
echo "数据库 student 选择成功! <BR>";
//创建表 t_student
$sql="CREATE TABLE t_student(xh CHAR(11) NOT NULL,";
$sql.="xm VARCHAR(12) NOT NULL,xb CHAR(2) NOT NULL,";
$sql.="bjdm CHAR(9) NOT NULL,csrq DATE,";
$sql.="PRIMARY KEY(xh))";
$sql.="ENGINE=MyISAM DEFAULT CHARSET=gb2312";
mysql_query($sql,$link) or die("表 t_student 创建失败! <BR>");
echo "表 t_student 创建成功! <BR>";
//创建表 t_class
$sql="CREATE TABLE t_class(bjdm CHAR(9) NOT NULL,";
$sql.="bjjc VARCHAR(15) NOT NULL,bjqc VARCHAR(30) NOT NULL,";
$sql.="PRIMARY KEY(bjdm))";
$sql.="ENGINE=MyISAM DEFAULT CHARSET=gb2312";
mysql_query($sql,$link) or die("表 t_class 创建失败! <BR>");
echo "表 t_class 创建成功! <BR>";
?>
```

该程序的运行结果如图 9.6 所示。

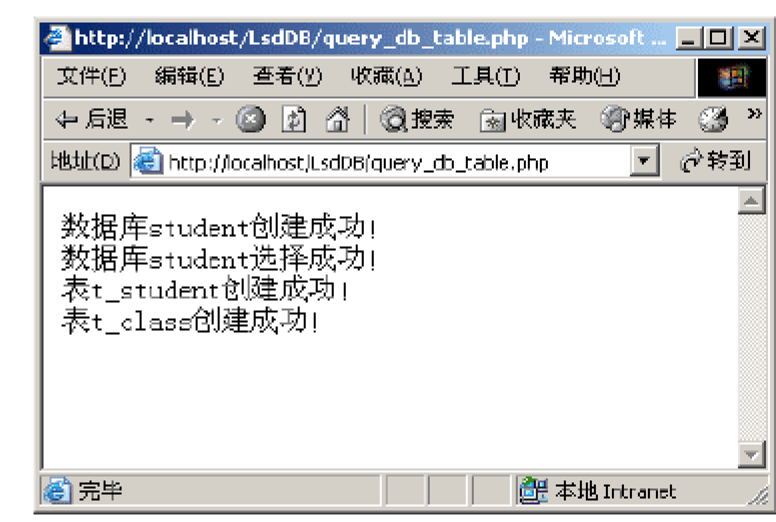


图 9.6 程序 query_db_table.php 的运行结果

4. 记录的检索与处理

为检索表中的记录，只需执行相应的 SELECT 语句即可。为处理检索到的记录，只需利用 mysql_query()函数在执行 SELECT 语句后所返回的结果标识符，并调用相应的处理函数即可。

mysql_query()函数所返回的结果标识符，通常又称之为结果集，代表了相应检索语句的检索结果。每个结果集都有一个记录指针，所指向的记录即为当前记录。在初始状态下，结果集的当前记录就是第一个记录。为灵活处理结果集中的有关记录，PHP 提供了一系列的处理函数，包括结果集中记录的读取、指针的定位以及记录集的释放等。

要读取结果集中的记录，可调用 mysql_fetch_array()、mysql_fetch_row() 或 mysql_fetch_assoc()函数。它们的语法格式为：

```
mysql_fetch_array(result[,type])
mysql_fetch_row(result)
mysql_fetch_assoc(result)
```

参数：result（resource 型）用于指定相应的结果集（或结果标识符）；type（int 型）用于指定函数返回值的形式，其有效取值为 PHP 常量 MYSQL_ASSOC、MYSQL_NUM 或 MYSQL_BOTH，默认值为 MYSQL_BOTH。

返回值：array 型。若成功（即读取到当前记录），则返回一个由结果集当前记录所生成的数组（每个字段的值保存到相应的元素中），并自动将记录指针指向下一个记录。若失败（即没有读取到记录），则返回 FALSE。

在调用 mysql_fetch_array()函数时，若以 MYSQL_NUM 作为第二个参数，则其功能与 mysql_fetch_row()函数的功能是一样的，所返回的数组为数字索引方式数组，只能以相应的序号（从 0 开始）作为元素的下标进行访问；若以 MYSQL_ASSOC 作为第二个参数，则其功能与 mysql_fetch_assoc()函数的功能是一样的，所返回的数组为关联索引方式数组，只能以相应的字段名（若指定了别名，则为相应的别名）作为元素的下标进行访问；若未指定第二个参数，或以 MYSQL_BOTH 作为第二个参数，则返回的数组为数字索引方式与关联索引方式数组，既能以序号为元素的下标进行访问，也能以字段名为元素的下标进行访问。由此可见，mysql_fetch_array()函数完全包含了 mysql_fetch_row()与 mysql_fetch_assoc()函数的功能。因此，在实际编程中，mysql_fetch_array()函数是最为常用的。

结果集被处理完毕后，为了及时释放其所占用的内存空间，可调用 mysql_free_result()函数。其语法格式为：

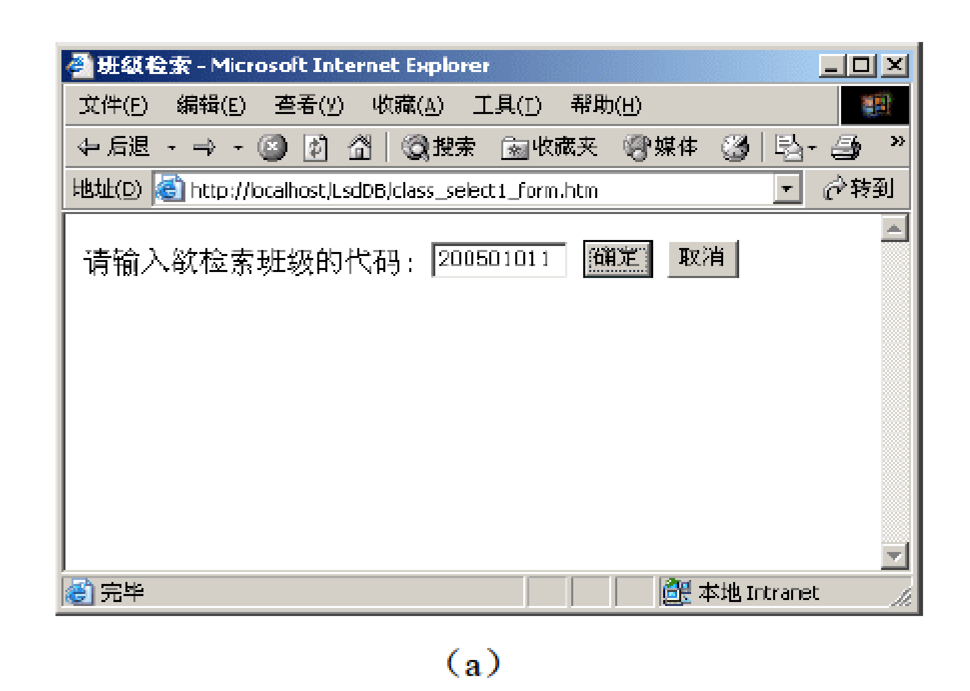
```
mysql_free_result(result)
```

参数：result（resource 型）用于指定相应的结果集（或结果标识符）。

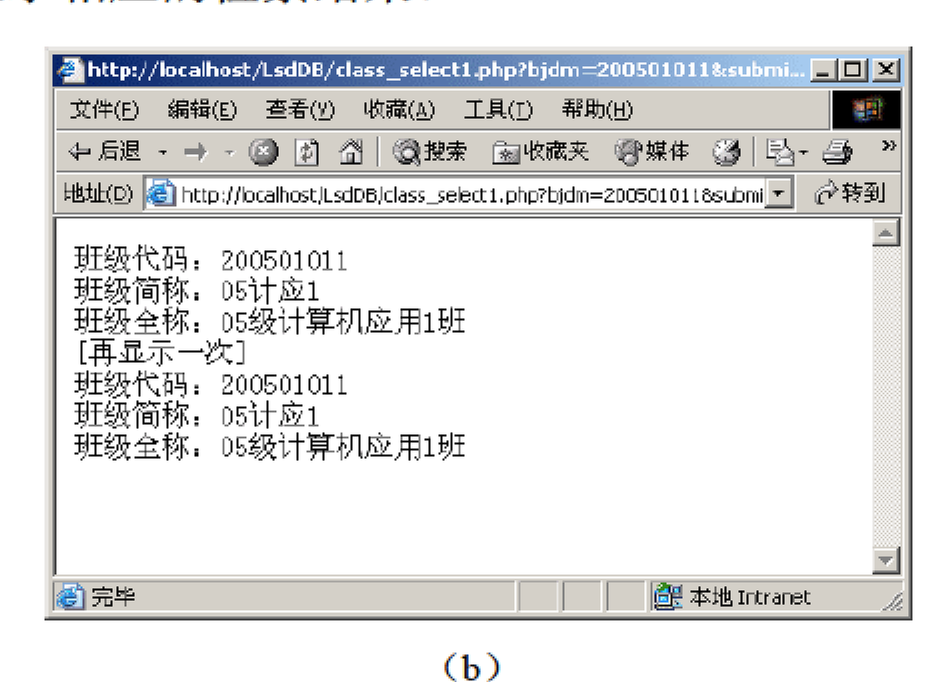
返回值：bool 型。若执行成功，则返回 TRUE；否则，返回 FALSE。

实际上，在程序执行结束后，结果集所占用的内存空间会自动被释放掉。因此，在 PHP 程序中，通常无须调用 mysql_free_result()函数。

【例 9.9】 班级记录的精确检索。如图 9.7 所示，先在班级检索表单中输入班级的代码，再单击“确定”按钮，即可开始检索并显示相应的检索结果。



（a）



（b）

图 9.7　班级记录的检索

（1）班级检索表单 class_select1_form.htm。

```
<html>
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>班级检索</title>
</head>
<body>
<form action="class_select1.php" method="get">
  请输入欲检索班级的代码：
  <input name="bjdm" type="text" id="bjdm" size="9" maxlength="9">
  <input name="submit" type="submit" value="确定">
  <input name="reset" type="reset" value="取消">
</form>
</body>
</html>
```

（2）班级检索程序 class_select1.php。

```
<?php
$bjdm=trim($bjdm);
if ($bjdm=="")  {
   echo "班级代码不能为空！";
   die();
}
$link=mysql_connect("localhost","root","123456")
  or die("数据库服务器连接失败！<BR>");
  mysql_select_db("student",$link) or die("数据库选择失败！<BR>");
mysql_query("set names 'gbk'");
$sql="select bjdm,bjjc,bjqc from t_class where bjdm='$bjdm'";
$result=mysql_query($sql,$link);
$row = mysql_fetch_array($result);
//$row = mysql_fetch_row($result);
//$row = mysql_fetch_assoc($result);
if (!$row)   {
    echo "无此班级代码!";
    die();
}
$bjdm=$row['bjdm'];
$bjjc=$row['bjjc'];
$bjqc=$row['bjqc'];
echo "班级代码: ".$bjdm."<BR>";
echo "班级简称: ".$bjjc."<BR>";
echo "班级全称: ".$bjqc."<BR>";
echo "[再显示一次]<BR>";
$bjdm=$row[0];
$bjjc=$row[1];
$bjqc=$row[2];
echo "班级代码: ".$bjdm."<BR>";
echo "班级简称: ".$bjjc."<BR>";
```

```
echo "班级全称: ".$bjqc."<BR>";
mysql_free_result($result);
?>
```


除了 mysql_fetch_array()、mysql_fetch_row()与 mysql_fetch_assoc()函数以外，要读取结果集中的记录，还可以使用 mysql_fetch_object()函数。其语法格式为：

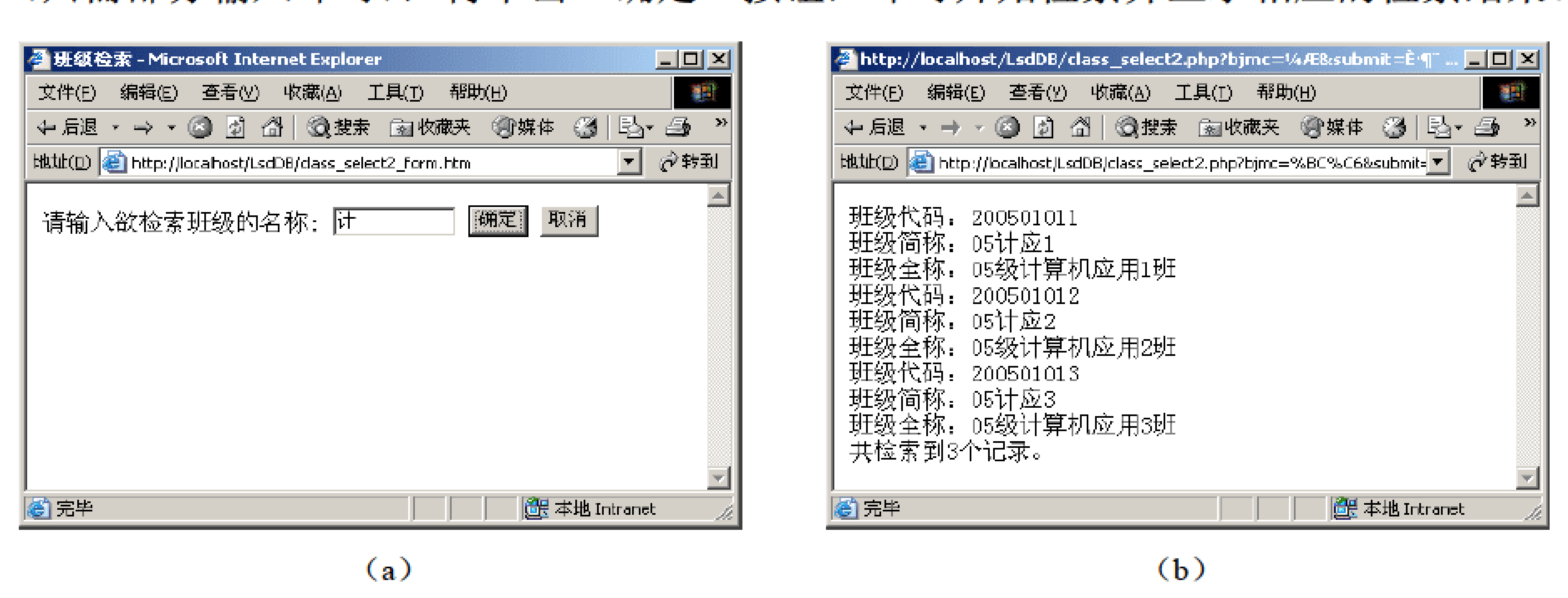
```
mysql_fetch_object(result)
```

参数：result（resource 型）用于指定相应的结果集（或结果标识符）。

返回值：object 型。若成功（即读取到当前记录），则返回一个由结果集当前记录所生成的对象（每个字段的值保存到相应的属性中），并自动将记录指针指向下一个记录；若失败（即没有读取到记录），返回 FALSE。

在 mysql_fetch_object()函数所返回的对象中，各属性的名称即为相应的字段名。在访问对象的属性时，应使用运算符“->”。

【例 9.10】 班级记录的模糊检索。如图 9.8 所示，先在班级检索表单中输入班级的名称（只需部分输入即可），再单击“确定”按钮，即可开始检索并显示相应的检索结果。



（a）　　（b）

图 9.8　班级记录的检索

（1）班级检索表单 class_select2_form.htm。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>班级检索</title>
</head>
<body>
<form action="class_select2.php" method="get">
  请输入欲检索班级的名称：
  <input name="bjmc" type="text" id="bjmc" size="10" maxlength="30">
  <input name="submit" type="submit" value="确定">
  <input name="reset" type="reset" value="取消">
</form>
</body>
</html>
```

（2）班级检索程序 class_select2.php。

```
<?php
$bjmc="%".trim($bjmc)."%";
$link=mysql_connect("localhost","root","123456")
    or die("数据库服务器连接失败！<BR>");
mysql_select_db("student",$link)
    or die("数据库选择失败！<BR>");
mysql_query("set names 'gbk'");
$sql="select bjdm,bjjc,bjqc from t_class";
$sql.=" where bjjc like '$bjmc' or bjqc like '$bjmc'";
$sql.=" order by bjdm";
$result=mysql_query($sql,$link);
$rows=0;
while($row=mysql_fetch_object($result))
{
  $rows=$rows+1;
  $bjdm=$row->bjdm;
  $bjjc=$row->bjjc;
  $bjqc=$row->bjqc;
  echo "班级代码: ".$bjdm."<BR>";
  echo "班级简称: ".$bjjc."<BR>";
  echo "班级全称: ".$bjqc."<BR>";
}
if ($rows==0)
  echo "没有满足指定条件的记录！";
else
  echo "共检索到".$rows."个记录。";
  mysql_free_result($result);
?>
```

有时候，需要获知结果集的记录数与字段数。为此，可使用 mysql_num_rows()与 mysql_num_fields()函数。它们的语法格式为：

```
mysql_num_rows(result)
mysql_num_fields(result)
```

参数：result（resource 型）用于指定相应的结果集（或结果标识符）。

返回值：int 型。mysql_num_rows()函数返回结果集的记录数，mysql_num_fields()函数返回结果集的字段数。

必要时，还可在结果集内随意移动记录的指针，也就是将记录指针直接指向某个记录。为此，需使用 mysql_data_seek()函数。其语法格式为：

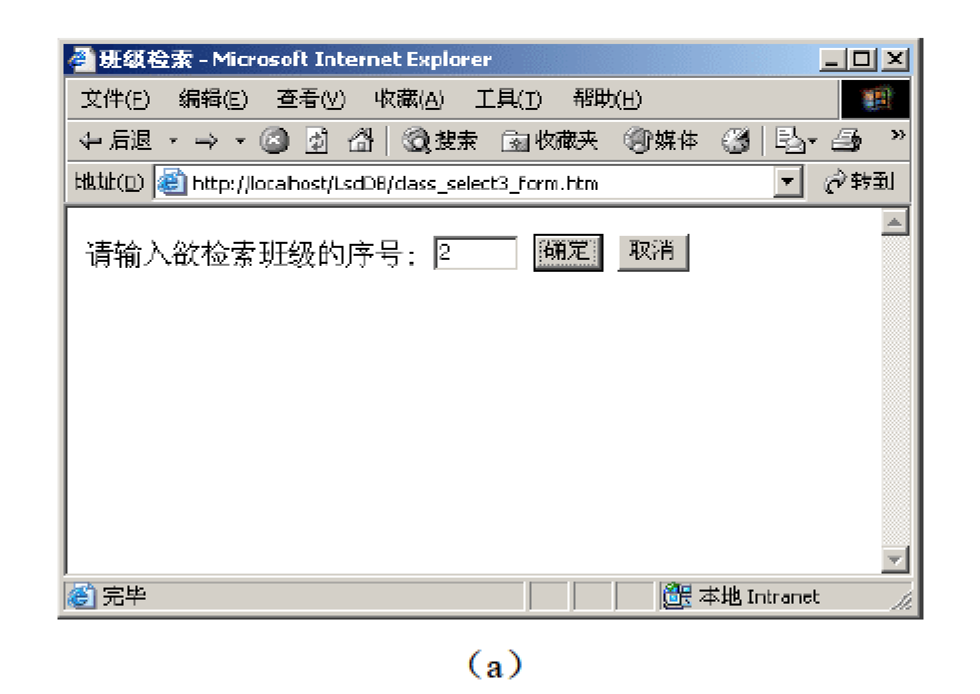
```
mysql_data_seek(result,row)
```

参数：result（resource 型）用于指定相应的结果集（或结果标识符）；row（int 型）用

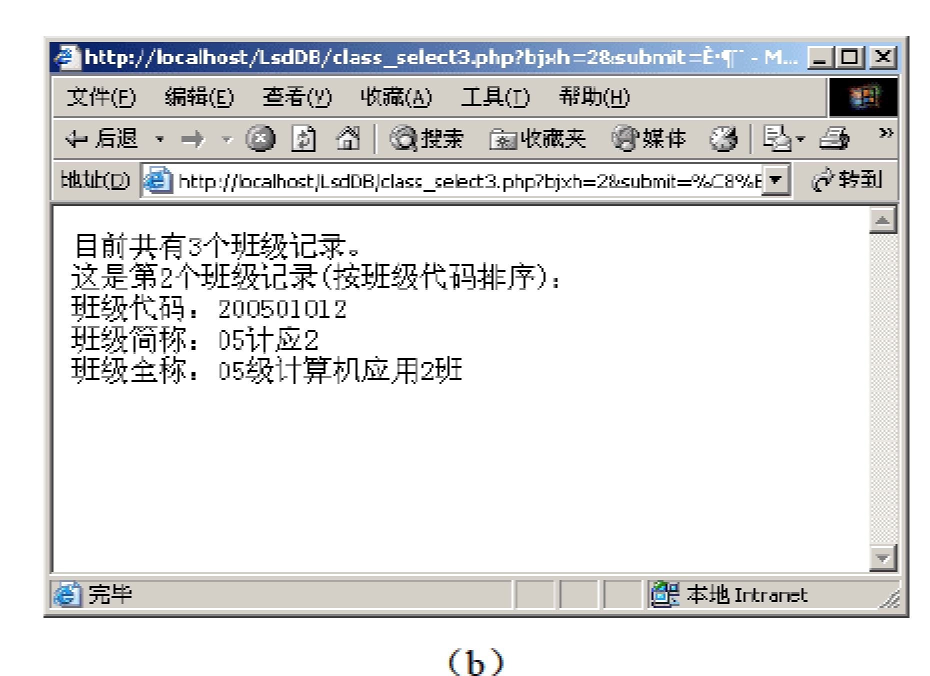
于指定记录指针所要指向的记录的序号（从 0 开始）。

返回值：bool 型。若执行成功，则返回 TRUE；否则，返回 FALSE。

【例 9.11】 班级记录的检索（按序号进行检索）。如图 9.9 所示，先在班级检索表单中输入班级的序号，再单击“确定”按钮，即可开始检索并显示相应的检索结果。



（a）



（b）

图 9.9　班级记录的检索

（1）班级检索表单 class_select3_form.htm。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>班级检索</title>
</head>
<body>
<form action="class_select3.php" method="get">
  请输入欲检索班级的序号：
  <input name="bjxh" type="text" id="bjxh" size="5" maxlength="5">
  <input name="submit" type="submit" value="确定">
  <input name="reset" type="reset" value="取消">
</form>
</body>
</html>
```

（2）班级检索程序 class_select3.php。

```
<?php
$link=mysql_connect("localhost","root","123456")
  or die("数据库服务器连接失败！<BR>");
mysql_select_db("student",$link) or die("数据库选择失败！<BR>");
mysql_query("set names 'gbk'");
$sql="select bjdm,bjjc,bjqc from t_class";
$sql=$sql." order by bjdm";
$result=mysql_query($sql,$link);
$rows=mysql_num_rows($result);
if ($rows==0)  {
```

```
  echo "目前还没有班级记录！";
  die();
}
echo "目前共有".$rows."个班级记录。<BR>";
if ($bjxh<1)
  $bjxh=1;
if ($bjxh>$rows)
  $bjxh=$rows;
mysql_data_seek($result,$bjxh-1);
$row = mysql_fetch_array($result);
echo "这是第".$bjxh."个班级记录(按班级代码排序): <BR>";
echo "班级代码: ".$row['bjdm']."<BR>";
echo "班级简称: ".$row['bjjc']."<BR>";
echo "班级全称: ".$row['bjqc']."<BR>";
mysql_free_result($result);
?>
```

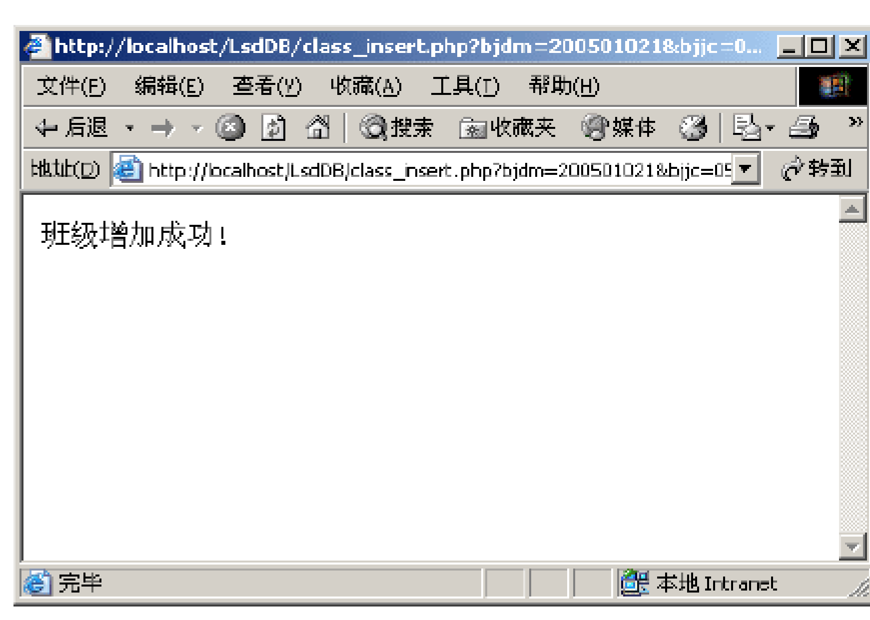
5．记录的增加、修改与删除

为增加记录，只需执行相应的 INSERT 语句即可；为修改记录，只需执行相应的 UPDATE 语句即可；为删除记录，只需执行相应的 DELETE 语句即可。

【例 9.12】 班级记录的增加。如图 9.10 所示，先在班级增加表单中输入班级的代码、简称与全称，再单击“确定”按钮，即可将班级记录插入至班级表 t_class 中（如果所输入的班级代码没有重复的话）。



(a)



(b)

图 9.10　班级记录的增加

（1）班级增加表单 class_insert_form.htm。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>班级增加</title>
</head>
<body>
<form action="class_insert.php" method="get">
```

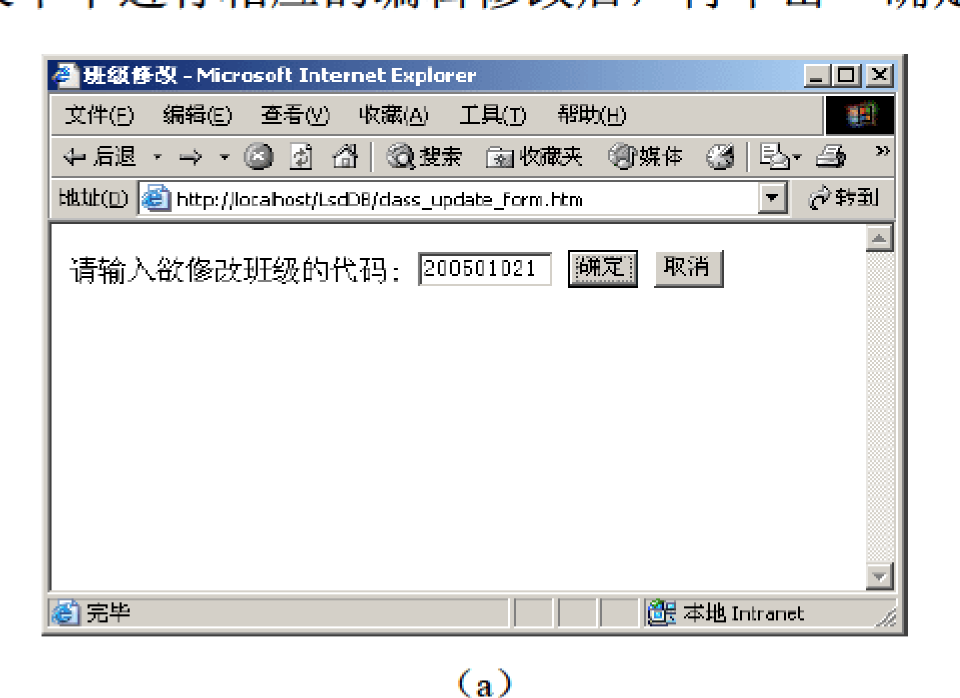
```
  <div align="center">班级增加</div> <br>
  <table width="300" border="1" align="center">
  <tr> <td width="85">班级代码: </td>
    <td width="199">
<input name="bjdm" type="text" size="9" maxlength="9"></td>
  </tr>
  <tr> <td>班级简称: </td>
    <td><input name="bjjc" type="text" size="15" maxlength="15"></td>
  </tr>
  <tr> <td>班级全称: </td>
         <td><input name="bjqc" type="text" size="20" maxlength="30"></td>
  </tr>
  </table>
  <br>
  <div align="center">
    <input name="submit" type="submit" value="确定">
    <input name="reset" type="reset" value="取消">
  </div>
</form>
</body>
</html>
```

（2）班级增加程序 class_insert.php。

```
<?php
$bjdm=trim($bjdm);
$bjjc=trim($bjjc);
$bjqc=trim($bjqc);
if ($bjdm==""||$bjjc==""||$bjqc=="")  {
   echo "班级代码及其简称与全称均不能为空! ";
   die();
}
$link=mysql_connect("localhost","root","123456")
  or die("数据库服务器连接失败! <BR>");
mysql_select_db("student",$link) or die("数据库选择失败! <BR>");
mysql_query("set names 'gbk'");
$sql="select bjdm from t_class where bjdm='$bjdm'";
$result=mysql_query($sql,$link);
$row = mysql_fetch_array($result);
if ($row)   {
    echo "此班级代码已经存在!";
    die();
 }
$sql="insert into t_class(bjdm,bjjc,bjqc)";
$sql=$sql." values('$bjdm','$bjjc','$bjqc')";
if (mysql_query($sql,$link))
```

```
    echo "班级增加成功!";
else
    echo '班级增加失败!';
?>
```

【例 9.13】 班级记录的修改。如图 9.11 所示，先在班级修改表单中输入班级的代码，再单击“确定”按钮进行查找，并将找到的班级记录显示在班级编辑表单中；在班级编辑表单中进行相应的编辑修改后，再单击“确定”按钮，即可完成相应的修改操作。



(a)



(b)

(c)

图 9.11　班级记录的修改

(1) 班级修改表单 class_update_form.htm。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>班级修改</title>
</head>
<body>
<form action="class_update_edit.php" method="get">
  请输入欲修改班级的代码：
  <input name="bjdm" type="text" size="9" maxlength="9">
  <input name="submit" type="submit" value="确定">
  <input name="reset" type="reset" value="取消">
```

```
</form>
</body>
</html>
```


（2）班级编辑程序 class_update_edit.php。

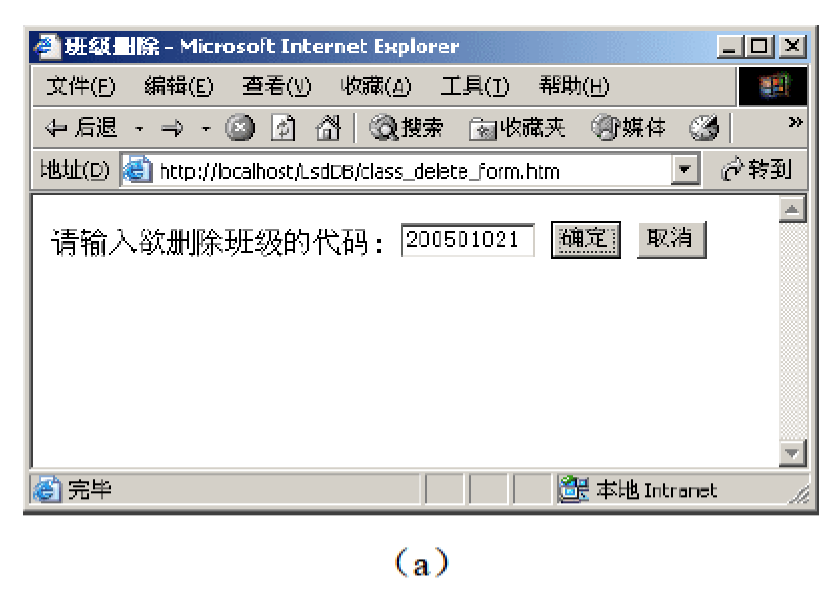
```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>班级编辑</title>
</head>
<body>
<?php
$bjdm=trim($bjdm);
if ($bjdm=="")  {
  echo "班级代码不能为空! ";
  die();
}
$link=mysql_connect("localhost","root","123456")
    or die("数据库服务器连接失败! <BR>");
mysql_select_db("student",$link)
    or die("数据库选择失败! <BR>");
mysql_query("set names 'gbk'");
$sql="select bjdm,bjjc,bjqc from t_class where bjdm='$bjdm'";
$result=mysql_query($sql,$link);
$row = mysql_fetch_array($result);
if (!$row)   {
  echo "无此班级代码!";
  die();
}
$bjdm=$row['bjdm'];
$bjjc=$row['bjjc'];
$bjqc=$row['bjqc'];
?>
<form action="class_update.php" method="get">
  <div align="center">班级编辑</div>  <br>
  <table width="300" border="1" align="center">
  <tr><td width="85">班级代码: </td>
    <td width="199"> <input name="bjdm" type="text" value="<?php echo
$bjdm; ?>" size="9" maxlength="9"></td>
  </tr>
  <tr>  <td>班级简称: </td>
    <td><input name="bjjc" type="text" value="<?php echo $bjjc; ?>" size="15"
maxlength="15"></td>
  </tr>
  <tr>  <td>班级全称: </td>
```

```
    <td><input name="bjqc" type="text" value="<?php echo $bjqc; ?>" size="20"
    maxlength="30"></td>
  </tr>
  </table>
  <input name="bjdm0" type="hidden" value="<?php echo $bjdm; ?>">
  <br>
  <div align="center">
    <input name="submit" type="submit" value="确定">
    <input name="reset" type="reset" value="取消">
  </div>
</form>
</body>
</html>
```

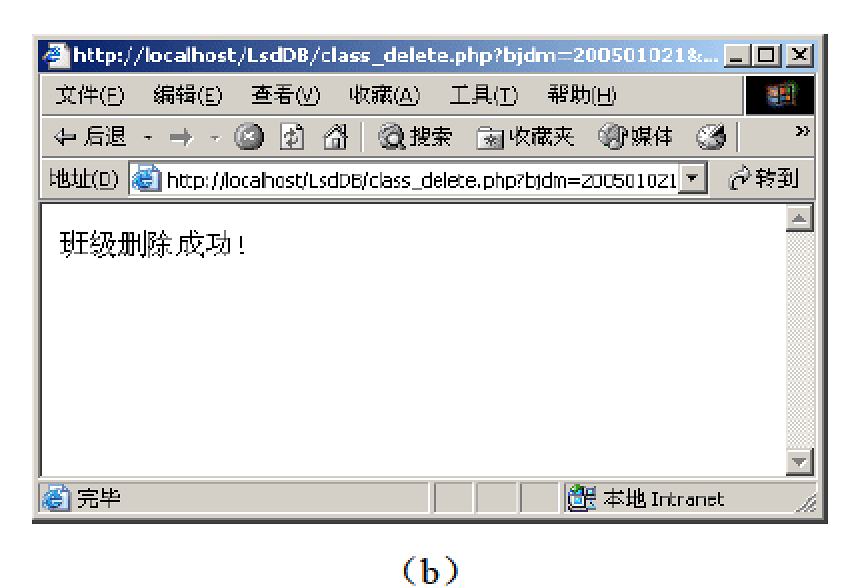
（3）班级修改程序 class_update.php。

```
<?php
$bjdm=trim($bjdm);
$bjjc=trim($bjjc);
$bjqc=trim($bjqc);
$bjdm0=trim($bjdm0);
if ($bjdm==""||$bjjc==""||$bjqc=="")  {
   echo "班级代码及其简称与全称均不能为空！";
   die();
}
$link=mysql_connect("localhost","root","123456")
  or die("数据库服务器连接失败！<BR>");
mysql_select_db("student",$link) or die("数据库选择失败！<BR>");
mysql_query("set names 'gbk'");
if ($bjdm<>$bjdm0)  {
  $sql="select bjdm from t_class where bjdm='$bjdm'";
  $result=mysql_query($sql,$link);
  $row = mysql_fetch_array($result);
  if ($row)  {
     echo "此班级代码已经存在!";
     die();
   }
 }
$sql="update t_class set bjdm='$bjdm',bjjc='$bjjc' ";
$sql=$sql." ,bjqc='$bjqc' where bjdm='$bjdm0'";
if (mysql_query($sql,$link))
  echo "班级修改成功!";
else
  echo '班级修改失败!';
?>
```

【例 9.14】 班级记录的删除。如图 9.12 所示，先在班级删除表单中输入班级的代码，再单击“确定”按钮，即可将相应的班级记录删除（如果有的话）。



（a）



（b）

图 9.12　班级记录的删除

（1）班级删除表单 class_delete_form.htm。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>班级删除</title>
</head>
<body>
<form action="class_delete.php" method="get">
  请输入欲删除班级的代码：
  <input name="bjdm" type="text" size="9" maxlength="9">
  <input name="submit" type="submit" value="确定">
  <input name="reset" type="reset" value="取消">
</form>
</body>
</html>
```

（2）班级删除程序 class_delete.php。

```
<?php
$bjdm=trim($bjdm);
if ($bjdm=="")  {
   echo "班级代码不能为空！";
   die();
}
$link=mysql_connect("localhost","root","123456")
  or die("数据库服务器连接失败！<BR>");
mysql_select_db("student",$link) or die("数据库选择失败！<BR>");
mysql_query("set names 'gbk'");
$sql="select bjdm from t_class where bjdm='$bjdm'";
$result=mysql_query($sql,$link);
$row = mysql_fetch_array($result);
```

```
if (!$row) {
   echo "无此班级代码!";
   die();
 }
$sql="delete from t_class where bjdm='$bjdm'";
if (mysql_query($sql,$link))
  echo "班级删除成功!";
else
  echo '班级删除失败!';
?>
```

9.2.4 关闭与数据库服务器的连接

对数据库的操作执行完毕后，应及时关闭与数据库服务器的连接，以释放其所占用的系统资源。在 PHP 中，为关闭由 mysql_connect()函数所建立的与 MySQL 数据库服务器的连接，可使用 mysql_close()函数。其语法格式为：

```
mysql_close([link_identifier])
```

参数：link_identifier（resource 型）为连接标识号，用于指定相应的与 MySQL 数据库服务器的连接。若未指定 link_identifier，则关闭上一个打开的连接。

返回值：bool 型。若执行成功，则返回 TRUE；否则，返回 FALSE。

【例 9.15】 关闭与数据库服务器的连接示例（close.php）。

```
<?php
$link=mysql_connect("localhost","root","123456")
    or die("无法建立与服务器的连接！");
echo "已成功建立与服务器的连接！<BR>";
mysql_select_db("student",$link)
    or die("无法选择 student 数据库！<BR>");
echo "已成功选择 student 数据库！<BR>";
mysql_close($link) or die("无法关闭与服务器的连接！");
echo "已成功关闭与服务器的连接！<BR>";
?>
```

由 mysql_connect()函数所建立的与 MySQL 数据库服务器的连接是一种非持久连接，可在程序执行后自动关闭。因此，在 PHP 程序中，通常无须调用 mysql_close()函数。此外，调用 mysql_close()函数不会关闭由 mysql_pconnect()函数所建立的持久连接。

9.3 使用 mysqli 函数库进行数据库编程

在 PHP 5 中，除了 mysql 函数库以外，还可以使用 mysqli 函数库访问 MySQL 数据库（MySQL 4.0 及以上版本）。为了使用 mysqli 函数库，需在 PHP 5 的配置文件 php.ini 中将“;extension=php_mysqli.dll”修改为“extension=php_mysqli.dll”（即删掉该选项前面的注释

符号“;”)，然后再重新启动 Web 服务器（如 Apache 等）。

mysqli 函数库的功能类似于 mysql 函数库，并有所增强，且支持两种使用方式——面向过程的使用方式与面向对象的使用方式。

若采用面向过程的方式，则 mysqli 函数库的用法与 mysql 函数库的用法是基本一致的，只是相应函数的名称有所不同罢了。在 mysql 函数库中，各函数的名称均以“mysql_”为前缀。而在 mysqli 函数库中，各函数的名称则以“mysqli_”为前缀。如以下示例。

```
$link=mysqli_connect("localhost","root","");  //建立连接
if (!$link)  {
    echo "服务器连接失败！<BR>";
    echo "错误编号: ".mysqli_connect_errno()."<BR>";
    echo "错误信息: ".mysqli_connect_error()."<BR>";
    die();
}
⋮
mysqli_close($link);  //关闭连接
```

若采用面向对象的方式，则 mysqli 函数库就相当于 mysqli 类与 result 类，而 mysqli 函数库中的函数就相当于 mysqli 类与 result 类的方法或属性。但作为 mysqli 类与 result 类中的方法与属性，其名称就无须以“mysqli_”作为前缀了。特别地，用于建立连接的 mysqli_connect()函数在 mysqli 类中已成为构造函数_ _construct()。如以下示例。

```
$link=new mysqli("localhost","root","");  //建立连接
if (mysqli_connect_errno())
{
    echo "服务器连接失败！<BR>";
    echo "错误编号: ".mysqli_connect_errno()."<BR>";
    echo "错误信息: ".mysqli_connect_error()."<BR>";
    die();
}
⋮
$link->close();  //关闭连接
```

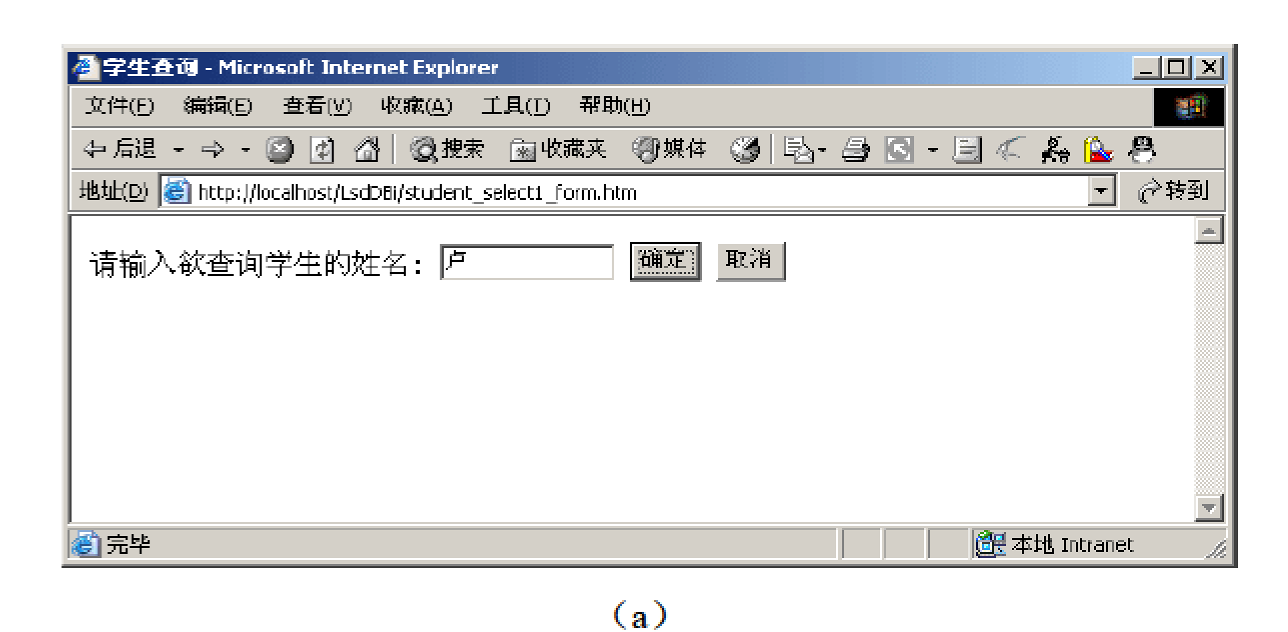
关于 mysqli 函数库的具体用法，请参阅 PHP 5 的帮助手册，在此不再详述。下面，将通过两个典型的实例，分别说明 mysqli 函数库的面向过程使用方法与面向对象使用方法，请注意对比掌握。

【例 9.16】 按姓名查询学生。如图 9.13 所示，先在学生查询表单中输入欲查询学生的姓名（只需部分输入即可），再单击“确定”按钮，即可以分页的形式显示相应的查询结果。在查询结果中，若单击“详情”链接，即可打开相应的窗口显示学生的详细信息；若单击“修改”或“删除”链接，则可实现相应记录的修改或删除操作（在此暂不实现这两项功能）。

（1）学生查询表单 student_select1_form.htm。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
```

```
<title>学生查询</title>
</head>
<body>
<form action="student_select1.php" method="get">
  请输入欲查询学生的姓名：
  <input name="xm" type="text" id="xm" size="12" maxlength="12">
  <input name="submit" type="submit" value="确定">
  <input name="reset" type="reset" value="取消">
</form>
</body>
</html>
```



(a)

学生查询结果 - Microsoft Internet Explorer

http://localhost/LsdDB/student_select1.php?xm=%C2%AC&submit=%C8%B7%B6%A8

学生查询结果

学号	姓名	性别	班级	出生日期	操作
20050101101	卢铭	男	05计应1	1986-06-01	详情 修改 删除
20050101104	卢强	男	05计应1	1987-02-16	详情 修改 删除

[第1页/共2页] 下一页 尾页 [共找到3个记录]

(b)



(c)

图 9.13　按姓名查询学生

（2）学生查询结果程序 student_select1.php（以面向过程方式使用 mysqli 函数库）。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>学生查询结果</title>
</head>
<body>
<?php
$xm=trim($xm);
if ($xm=="")  {
  echo "请输入欲查询学生的姓名！";
```

```
      die();
    }
    $xm0="%".$xm."%";
    $link=mysqli_connect("localhost","root","123456")
      or die("数据库服务器连接失败！<BR>");
    mysqli_select_db($link,"student") or die("数据库选择失败！<BR>");
    mysqli_query($link,"set names 'gbk'");
    $sql="select xh,xm,xb,t_class.bjjc as bjjc,csrq from t_student,t_class";
    $sql.=" where t_student.bjdm=t_class.bjdm and t_student.xm like '$xm0' order
by xh";
    $result=mysqli_query($link,$sql);
    $rows=mysqli_num_rows($result);  //总记录数
    if ($rows==0)  {
      echo "没有满足条件的记录！";
      die();
    }
    $pagesize=2;  //每页的记录数(在此暂设为 2，通常应设为 10)
    $pagecount=ceil($rows/$pagesize);  //总页数
    //$pageno 的值为当前页的页号
    if (!isset($pageno)||$pageno<1)
        $pageno=1;
    if ($pageno>$pagecount)
        $pageno=$pagecount;
    $offset=($pageno-1)*$pagesize;
    mysqli_data_seek($result,$offset);
    ?>
    <div align="center"><strong>学生查询结果</strong> </div>
    <table width="90%" border="1" align="center">
      <tr>
        <td><div align="center">学号</div></td>
        <td><div align="center">姓名</div></td>
        <td><div align="center">性别</div></td>
        <td><div align="center">班级</div></td>
        <td><div align="center">出生日期</div></td>
        <td><div align="center">操作</div></td>
      </tr>
    <?php
      $i=0;
      while($row=mysqli_fetch_array($result))
      {
    ?>
      <tr>
        <td><div align="center"><?php echo $row['xh']; ?></div></td>
        <td><div align="center"><?php echo $row['xm']; ?></div></td>
        <td><div align="center"><?php echo $row['xb']; ?></div></td>
```

```
        <td><div align="center"><?php echo $row['bjjc']; ?></div></td>
        <td><div align="center"><?php echo $row['csrq']; ?></div></td>
        <td><div align="center">
        <a href="student_detail1.php?xh=<?php echo $row['xh']; ?>" target="_
blank">详情</a>
        <a href="student_update1.php?xh=<?php echo $row['xh']; ?>" target=
"_blank">修改</a>
        <a href="student_delete1.php?xh=<?php echo $row['xh']; ?>" target="_
blank">删除</a>
        </div></td>
      </tr>
    <?php
      $i=$i+1;
      if ($i==$pagesize)
        break;
      }
      mysqli_free_result($result);
      mysqli_close($link);
    ?>
    </table>
    <div align="center">
    [第<?php echo $pageno; ?>页/共<?php echo $pagecount; ?>页]
    <?php
    $href=$PHP_SELF."?xm=".urlencode($xm);
    if ($pageno<>1){
    ?>
      <a href="<?php echo $href; ?>&pageno=1">首页</a>
      <a href="<?php echo $href; ?>&pageno=<?php echo $pageno-1; ?>">上一页</a>
    <?php
    }
    if ($pageno<>$pagecount){
    ?>
    <a href="<?php echo $href; ?>&pageno=<?php echo $pageno+1; ?>">下一页</a>
    <a href="<?php echo $href; ?>&pageno=<?php echo $pagecount; ?>">尾页</a>
    <?php
    }
    ?>
    [共找到<?php echo $rows; ?>个记录]
    </div>
    </body>
    </html>
```

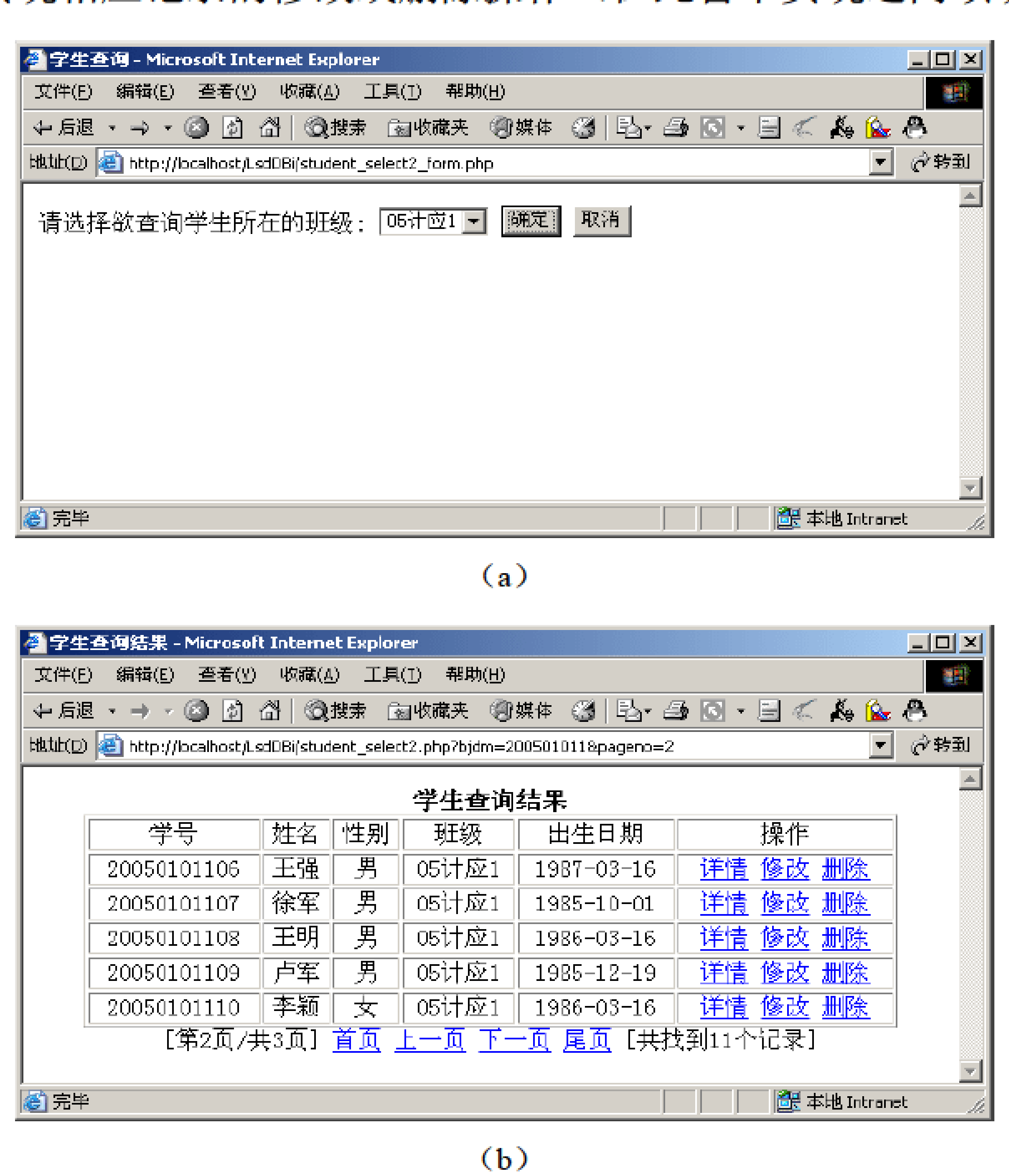
（3）学生详细信息程序 student_detail1.php（以面向过程方式使用 mysqli 函数库）。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
```

```
<title>学生详细信息</title>
</head>
<body>
<?php
$xh=trim($xh);
$link=mysqli_connect("localhost","root","123456")
  or die("数据库服务器连接失败! <BR>");
mysqli_select_db($link,"student") or die("数据库选择失败! <BR>");
mysqli_query($link,"set names 'gbk'");
$sql="select xh,xm,xb,t_class.bjjc as bjjc,csrq from t_student,t_class";
$sql=$sql." where t_student.bjdm=t_class.bjdm and t_student.xh='$xh'";
$result=mysqli_query($link,$sql);
$row=mysqli_fetch_array($result);
?>
<div align="center"><strong>学生详细信息</strong> </div>
<br>
<table width="350" border="1" align="center">
  <tr>
    <td width="100"><div align="right">学号: </div></td>
    <td><?php echo $row['xh']; ?> <div align="left"></div></td>
  </tr>
  <tr>
    <td><div align="right">姓名: </div></td>
    <td><?php echo $row['xm']; ?> <div align="left"></div></td>
  </tr>
  <tr>
    <td><div align="right">性别: </div></td>
    <td><?php echo $row['xb']; ?> <div align="left"></div></td>
  </tr>
  <tr>
    <td><div align="right">班级: </div></td>
    <td><?php echo $row['bjjc']; ?> <div align="left"></div></td>
  </tr>
  <tr>
    <td><div align="right">出生日期: </div></td>
    <td><?php echo $row['csrq']; ?> <div align="left"></div></td>
  </tr>
</table>
<?php
  mysqli_free_result($result);
  mysqli_close($link);
?>
<br>
<div align="center">
<a href="javascript:window.close()">[关闭窗口]</a></div>
</body>
</html>
```

【例 9.17】 按班级查询学生。如图 9.14 所示，先在学生查询表单中选择欲查询学生所

在的班级，再单击“确定”按钮，即可以分页的形式显示相应的查询结果。在查询结果中，若单击“详情”链接，即可打开相应的窗口显示学生的详细信息；若单击“修改”或“删除”链接，则可实现相应记录的修改或删除操作（在此暂不实现这两项功能）。



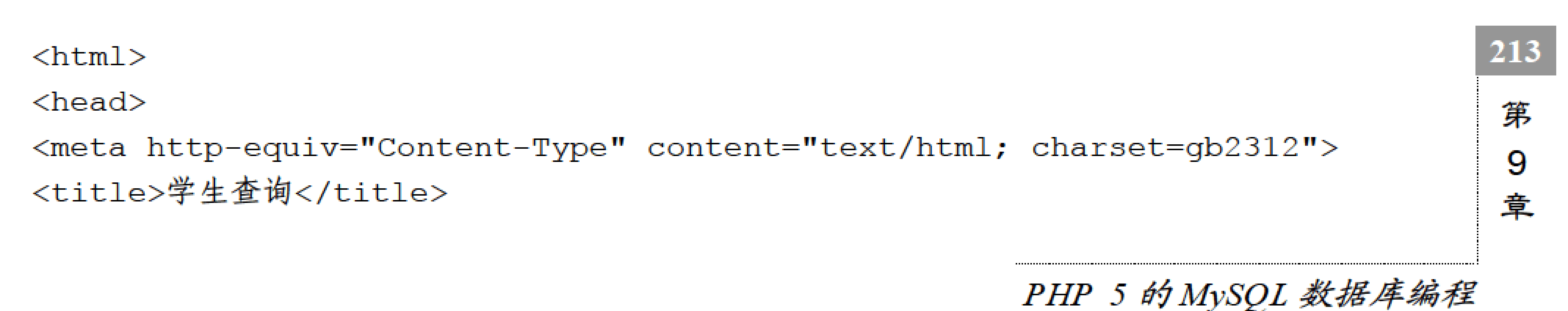
（a）

（b）

（c）

图 9.14　按班级查询学生

（1）学生查询表单 student_select2_form.php（以面向对象方式使用 mysqli 函数库）。



```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>学生查询</title>
```

```
</head>
<body>
<form action="student_select2.php" method="get">
  请选择欲查询学生所在的班级：
  <select name="bjdm" size="1">
<?php
  $link=new mysqli("localhost","root","123456");
  if (mysqli_connect_errno())  {
    echo "数据库服务器连接失败！<BR>";
    die();
  }
  $link->select_db("student") or die("数据库选择失败！<BR>");
  $link->query("set names 'gbk'");
  $sql="select bjdm,bjjc from t_class order by bjjc";
  $result=$link->query($sql);
  while($row=$result->fetch_object())  {
?>
  <option value="<?php echo $row->bjdm; ?>"><?php echo $row->bjjc;?></option>
<?php
  }
  $result->free();
  $link->close();
?>
  </select>
  <input name="submit" type="submit" value="确定">
  <input name="reset" type="reset" value="取消">
</form>
</body>
</html>
```

（2）学生查询结果程序 student_select2.php（以面向对象方式使用 mysqli 函数库）。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>学生查询结果</title>
</head>
<body>
<?php
  $bjdm=trim($bjdm);
  $link=new mysqli("localhost","root","123456");
  if (mysqli_connect_errno())  {
    echo "数据库服务器连接失败！<BR>";
    die();
  }
  $link->select_db("student") or die("数据库选择失败！<BR>");
```

```
  $link->query("set names 'gbk'");
  $sql="select xh,xm,xb,t_class.bjjc as bjjc,csrq from t_student,t_class";
  $sql.=" where t_student.bjdm=t_class.bjdm ";
  $sql.=" and t_student.bjdm='$bjdm' order by xh";
  $result=$link->query($sql);
  $rows=$result->num_rows;  //总记录数
  if ($rows==0)  {
    echo "没有满足条件的记录！";
    die();
  }
  $pagesize=5;  //每页的记录数(在此暂设为 5，通常应设为 10)
  $pagecount=ceil($rows/$pagesize);  //总页数
  //$pageno 的值为当前页的页号
  if (!isset($pageno)||$pageno<1)
    $pageno=1;
  if ($pageno>$pagecount)
    $pageno=$pagecount;
  $offset=($pageno-1)*$pagesize;
  $result->data_seek($offset);
?>
<div align="center"><strong>学生查询结果</strong> </div>
<table width="90%" border="1" align="center">
  <tr>
    <td><div align="center">学号</div></td>
    <td><div align="center">姓名</div></td>
    <td><div align="center">性别</div></td>
    <td><div align="center">班级</div></td>
    <td><div align="center">出生日期</div></td>
    <td><div align="center">操作</div></td>
  </tr>
<?php
  $i=0;
  while($row=$result->fetch_object())  {
?>
  <tr>
    <td><div align="center"><?php echo $row->xh; ?></div></td>
    <td><div align="center"><?php echo $row->xm; ?></div></td>
    <td><div align="center"><?php echo $row->xb; ?></div></td>
    <td><div align="center"><?php echo $row->bjjc; ?></div></td>
    <td><div align="center"><?php echo $row->csrq; ?></div></td>
    <td><div align="center">
    <a href="student_detail2.php?xh=<?php  echo  $row->xh;  ?>"  target="_
blank">详情</a>
    <a href="student_update2.php?xh=<?php echo $row->xh; ?>" target="_blank">
修改</a>
```

```
        <a href="student_delete2.php?xh=<?php echo $row->xh; ?>" target="_blank">
删除</a>
        </div></td>
      </tr>
    <?php
      $i=$i+1;
      if ($i==$pagesize)
        break;
      }
      $result->free();
      $link->close();
    ?>
    </table>
    <div align="center">
    [第<?php echo $pageno; ?>页/共<?php echo $pagecount; ?>页]
    <?php
    $href=$PHP_SELF."?bjdm=".urlencode($bjdm);
    if ($pageno<>1) {
    ?>
      <a href="<?php echo $href; ?>&pageno=1">首页</a>
      <a href="<?php echo $href; ?>&pageno=<?php echo $pageno-1; ?>">上一页</a>
    <?php
    }
    if ($pageno<>$pagecount) {
    ?>
    <a href="<?php echo $href; ?>&pageno=<?php echo $pageno+1; ?>">下一页</a>
    <a href="<?php echo $href; ?>&pageno=<?php echo $pagecount; ?>">尾页</a>
    <?php
    }
    ?>
    [共找到<?php echo $rows; ?>个记录]
    </div>
    </body>
    </html>
```

（3）学生详细信息程序 student_detail2.php（以面向对象方式使用 mysqli 函数库）。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>学生详细信息</title>
</head>
<body>
<?php
  $xh=trim($xh);
  $link=new mysqli("localhost","root","123456");
```

```
  if (mysqli_connect_errno()) {
    echo "数据库服务器连接失败！<BR>";
    die();
  }
  $link->select_db("student")  or die("数据库选择失败！<BR>");
  $link->query("set names 'gbk'");
  $sql="select xh,xm,xb,t_class.bjjc as bjjc,csrq from t_student,t_class";
  $sql=$sql." where t_student.bjdm=t_class.bjdm and t_student.xh='$xh'";
  $result=$link->query($sql);
  $row=$result->fetch_object()
?>
<div align="center"><strong>学生详细信息</strong> </div>
<br>
<table width="350" border="1" align="center">
  <tr>
    <td width="100"><div align="right">学号: </div></td>
    <td><?php echo $row->xh; ?> <div align="left"></div></td>
  </tr>
  <tr>
    <td><div align="right">姓名: </div></td>
    <td><?php echo $row->xm; ?> <div align="left"></div></td>
  </tr>
  <tr>
    <td><div align="right">性别: </div></td>
    <td><?php echo $row->xb; ?> <div align="left"></div></td>
  </tr>
  <tr>
    <td><div align="right">班级: </div></td>
    <td><?php echo $row->bjjc; ?> <div align="left"></div></td>
  </tr>
  <tr>
    <td><div align="right">出生日期: </div></td>
    <td><?php echo $row->csrq; ?> <div align="left"></div></td>
  </tr>
</table>
<?php
  $result->free();
  $link->close();
?>
<br>
<div align="center">
<a href="javascript:window.close()">[关闭窗口]</a></div>
</body>
</html>
```

实 验 9

1．请完成本章示例数据库与表的创建，并在表中输入相应的数据。

2．请使用 mysql 函数库，自行设计并实现学生记录的检索、增加、修改与删除功能。

3．请使用 mysqli 函数库，自行设计并实现学生记录的检索、增加、修改与删除功能。

习 题 9

1．请简述 PHP 中 MySQL 数据库编程的基本步骤。

2．在 PHP 中，如何建立与 MySQL 数据库服务器的连接？

3．在 PHP 中，如何关闭与 MySQL 数据库服务器的连接？

4．在 PHP 中，如何实现 MySQL 数据库的选择？

5．在 PHP 中，如何实现 MySQL 数据库的有关操作（如记录的检索、增加、修改与删除等)?

6．请简述在 PHP 5 中使用 mysqli 函数库访问 MySQL 数据库的基本方法。

第 10 章 会话和用户认证

本章导读

一般情况下，大多数 Web 站点都允许匿名访问，也就是说，任何连接到 Internet 的用户都可以直接进入该网站并查看其中的页面，用户无须进行身份验证就可以访问网站中的文件。然而，对于部分网站来说，并不是所有页面都能够匿名访问，比如网络学院的远程教学、付费信息、站点的管理区等，它们通常要强制用户提供自己的身份凭证。通常使用的身份凭证是用户名和密码。用户身份验证通过后将页面定向到受保护的页面。同时可以根据需要将用户的用户名、密码等特殊信息存储起来，以便用户访问其他受限制的页面时不需要重新输入身份凭证。这些就是本章要解决的问题。

本章介绍的内容包括三种重定向网页的方法，使浏览器访问某一网页时自动地跳转到另一个网页；使用 header()函数向浏览器发送各种 HTTP 标头信息；接下来介绍 HTTP 基本认证原理及其四种基本认证技术，摘要认证原理及实现技术；最后介绍使用 COOKIE 和 SESSION 保存、读取用户信息。这些技术都是 Web 应用安全性方面要加以考虑的因素。

10.1 网页重定向

在网页中，可以利用超链接把用户导航到另一个页面，但用户必须单击超链接才可以实现。有时候要求页面自动重定向到另一个页面，显示另一个页面内容。例如，用户没有登录而访问一个论坛时，就需要使页面自动地跳转到登录页面。网页重定向常用于这样的场合：一个已被广大用户熟悉的网页文件的位置发生了变化，应用网页重定向使老用户在无通知的情况下，仍然能够访问新位置中的网页。实现重定向网页的方法有三种：一是利用 PHP 的 header()函数，二是利用 HTML 的 META 标记，三是利用 JavaScript 脚本程序。其中第 2 种方法已在第 2 章的 2.2 节介绍，第 3 种方法已在第 3 章的 3.5 节介绍。这里，只介绍 header()函数的使用。因为 header()函数与 HTTP 协议的报头信息相关，下面先介绍 HTTP 报头，然后介绍 header()函数。

10.1.1 HTTP 协议报头

HTTP 协议是 Web 的通信协议，HTTP 协议的当前版本号是 1.1。在 1999 年 6 月批准的 RFC2616 文档中定义了 HTTP 协议的详细内容，其网址是 http://www.w3.org/Protocols/rfc2616/rfc2616。

HTTP 协议由两阶段组成，即请求和响应。浏览器和 Web 服务器之间的每个 HTTP 通信的请求消息或响应消息都包括 HTTP 头部和主体两部分。浏览器向服务器发送一个请

求，请求消息包含请求的方法、URI、协议版本、头部字段、一个空行和请求主体。服务器接收到请求后，给浏览器发送相应的响应信息，响应消息包括状态行、响应头部字段、一个空行和响应主体。其中，头部字段可以有一个或多个字段，每个头部字段由一个域名、冒号（:）和域值三部分组成。域名是大小写无关的，域值前可以添加任何数量的空格符。

传统的头部字段一定包含下面三种字段之一，并只能出现一次。

```
Location: xxxx:yyyy/zzzz
Content-Type: xxxx/yyyy
Status:  nnn  xxxxxx
```

其中，Location 字段表示向浏览器发送重定向代码，并显示指定 URL 地址的网页；Content-Type 字段指明发送给浏览器的 MIME 类型；Status 字段指明向浏览器发送的状态码。

10.1.2 PHP 的 header()函数

在 PHP 脚本语言中，header()函数的功能是用来向浏览器发送 HTTP 头部信息。其语法形式如下：

```
void header(string string [,bool replace[,int http_response_code]])
```

参数说明：

（1）string 参数是将被发送的 HTTP 头部字段，其格式为

```
域名:域值
```

例如，下面 PHP 程序向浏览器发送一个 HTTP 头部信息，告诉浏览器即将发送的内容是 HTML 文档，文档字符集为 UTF-8 编码。

```
<? header("Content-Type: text/html; charset= UTF-8"); ?>
```

该程序与以下 HTML 的 META 标记的作用相同。

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

（2）可选的 replace 参数指明是替换掉前一条类似的头标还是增加一条相同类型的头标。默认值为替换。但如果将其值设为 false，则可以强制发送多个同类头标。

（3）可选的 http_response_code 参数强制将 HTTP 响应代码设为指定值（此参数适用于 PHP 4.3.0 或以上版本）。

下面介绍 header()函数常用的几种应用。

1．重定向网页

header()函数最常见的应用之一是重定向网页。在脚本程序中，通过利用 header()函数，强制地向浏览器发送一个 HTTP 协议的 Location 头部字段，同时将一个 REDIRECT(302)状态码发送给浏览器，使得浏览器跳转到指定 URL 地址的页面。例如，下列程序使浏览器显示 www.php.net 网站的主页内容。

```
<?php
  Header("Location: http://www.php.net");
  exit;
?>
```

【例 10.1】 编写一个含有表单的网页文件 ex10_1.html，提交数据后调用文件名为 ex10_2.php 的 PHP 程序，由它检查表单的所有输入是否为空。

```
ex10_1.html 网页文件的内容为:
<html>
<head>
<title>用 header()函数重定向网页</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
</head>
<body>
<form action="ex10_2.php" method="post">
  姓名: <input type="text" name="yourname">
  <input type="submit" value="确定">
</form>
</body>
</html>
```

上述网页文件的表单数据提交后调用例 10.2 的 PHP 脚本程序。

【例 10.2】 下列程序用来检查表单的数据是否为空。如果是空值，则重新显示表单，要求重新输入数据。ex10_2.php 程序如下：

```
<?php
if (trim($_POST['yourname']) == ""){
    header("Location:ex10_1.html");
    exit;
}
echo "你的名字是 ".$_POST['yourname'];
?>
```

工作原理：程序中，使用系统数组$_POST 的$_POST['yourname'] 元素获取从浏览器传来的表单数据，如果是空的字符串，通过调用 header()函数，向浏览器发送 Location 头标，从而浏览器显示指定的 ex10_1.html 网页文件的内容。图 10.1 是访问 ex10_1.html 文件的示例，图 10.2 是单击图 10.1 中的“确定”按钮后，调用 ex10_2.php 程序的结果。



图 10.1 ex10_1.html 输入表单

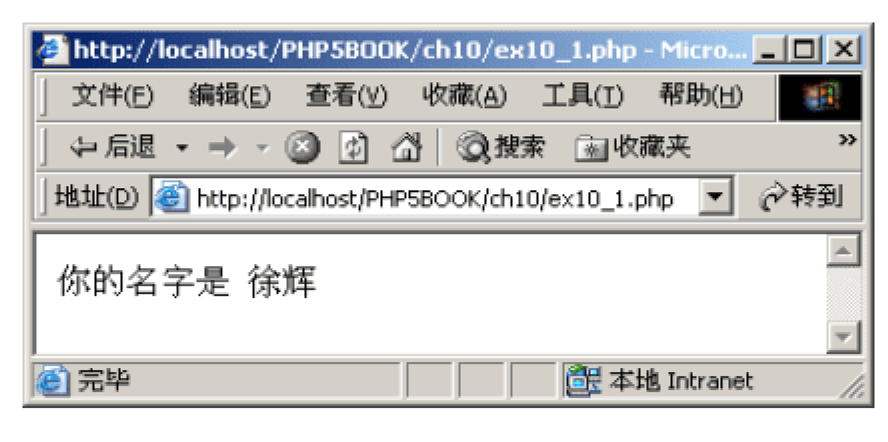


图 10.2 调用 ex10_2.php 的结果

2．向浏览器发送非 HTML 文档的内容

有时候需要向浏览器输出非 HTML 文档的其他格式类型的文件内容，例如，需要从数据库中读取记录内容，动态地生成一个 PDF 格式的文档，传送到浏览器上并显示。此时需要告诉浏览器，输出文件的类型是 PDF 类型，再传送 PDF 文档。利用 header()函数可以很好地解决这类问题。通过在 header()函数中声明输出内容的类型，可以输出其他类型的文件。以下例子实现向用户输出一个动态生成的 JPG 图形。

【例 10.3】 显示动态生成的 JPG 图形，EX10_3.html 网页文件内容如下。

```
<html>
<head>
<title>显示 PHP 动态生成的图形</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
</head>
<body>
<div align="left"><img src="ex10_4.php"> </div>
</body>
</html>
```

上述网页文件中图像标签<img src=…>所指的图形是通过执行 ex10_4.php 程序生成的，其程序内容如下。

【例 10.4】 输出动态图形的 PHP 程序（ex10_4.php），程序内容如下。

```
<?php
$path = "images/winter.jpg";   //图形的路径
try {
    if (is_file($path)){
        $file = fopen($path, 'rb');
         $f=fread($file,filesize($path));
         fclose($file);
         //用 header()函数输出一个.jpg 图形
         header("Content-type: image/jpeg");
         echo $f;
    } else {
         throw new exception("Sorry, file path is not valid!");
    }
} catch (exception $e){
    header("Content-type: image/jpeg");
    //创建一个动态的错误信息
    $animage = imagecreate(500, 100) or die("不能初始化 GD 库");
    $red = imagecolorallocate($animage, 255, 0, 0);
    $white = imagecolorallocate($animage, 255, 255, 255);
    imagefilledrectangle($animage, 0, 0, 500, 500, $white);
```

```
        imagestring($animage,5,2,5,$e->getmessage(),$red);
        imagejpeg($animage);
        imagedestroy($animage);
    }
    ?>
```

当浏览器访问 ex10_3.html 文件时，如果相对路径 images 下存在名为 winter.jpg 的图形文件，则显示与图 10.3 类似的图形文件内容；如果该图形文件不存在，则显示由英文“Sorry, file path is not valid! ”构成的图形，如图 10.4 所示。



图 10.3　图形存在时访问 ex10_3.html 的结果

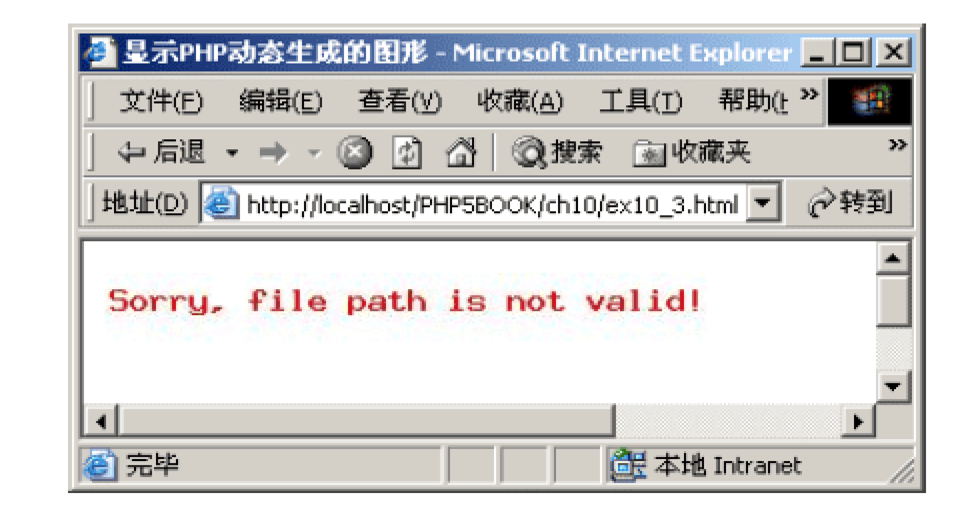


图 10.4　图形不存在时访问 ex10_3.html 的结果

提示：由于本程序使用了图像处理函数，因此，如果使用 Windows 平台的 Web 服务器环境，需要修改 PHP 的配置文件 php.ini，将 GD2 库的动态链接库文件 php_gd2.dll 作为一个扩展包含在 php.ini 中，指令行如下：

```
extension=php_gd2.dll
```

工作原理：在 ex10_4.php 程序中，利用 try…catch 语句对 try 分块的语句进行错误捕获，如果指定的图形文件不存在，那么执行 throw 语句，抛出一个异常，由 catch 分块进行异常处理。在 try 分块中，利用文件操作函数以二进制方式读取一个 winter.jpg 图形文件，然后利用 header()函数输出 HTTP 头标的内容类型“Content-type: image/jpeg”，指明将要输出的是一个图形，接着执行 echo 语句输出图形内容的值，从而在浏览器是显示一个图形，如图 10.3 所示。可以使用同样的代码，读取更多的其他文件作为二进制数据，然后按同样的方式，使用合适的内容类型，输出其他文件内容。表 10.1 列出了常见的内容类型。更多的内容类型详见 Apache 安装文件夹中 conf 子文件夹下的 mime.types 文件。

表 10.1　常用的文件格式内容类型

内容类型	描述说明
application/pdf	Adobe 的 PDF 类型，表示要输出 PDF 文档
application/msword	输出 Word 文档
application/excel	输出 Excel 文档
image/gif	输出 GIF 图形
image/png	输出 PNG 图形
application/octet-stream	输出一个 Zip 压缩文件
text/plain	输出文本文件

catch 分块进行异常处理，利用 imagecreate()图形处理函数创建一个$animage 动态图像对象，用 imagestring()函数生成由英文“Sorry, file path is not valid!”组成的图形，用 imagejpeg()函数输出生成的图形。

3．强制文件下载时显示“文件下载”对话框

一般情况下，可以直接在浏览器中输出各种不同类型的文件。在默认情况下，使用 header()函数输出其他文件类型时，该文件内容自动地显示在浏览器窗口中。如果需要下载该文件，可以用 header()函数强制浏览器显示“文件下载”对话框。下面例 10.5 的程序代码与上述例 10.4 的程序代码大部分相同，不同的是增加了一个 header()函数，强制用户下载文件。

【例 10.5】 强制用户下载文件的 PHP 程序（ex10_5.php），程序内容如下。

```
<?php
$path = "images/winter.jpg";
try {
    if (is_file($path)){
        $file = fopen($path, 'rb');
        $f=fread($file,filesize($path));
        fclose($file);
        $outputname = "myimage";
        header("Content-type: image/jpeg");
        //强制下载文件
        header("Content-disposition: attachment; filename=".$outputname.
        ".jpg");
        print $f;
    } else {
        throw new exception("Sorry, file path is not valid.");
    }
} catch (exception $e) {
    echo $e->getmessage();
}
?>
```

工作原理：程序中，实现文件下载的关键之处是第二个 header()函数，它向浏览器输出 HTTP 头标“content-disposition”。在该头标中将 content-disposition 字段的值设置为 attachment，指定 filename 属性值为一个要保存的文件名，强制浏览器下载文件而不是显示文件内容。利用第一个 header()函数输出 HTTP 头标“content-type”，强制浏览器按 JPG 图形类型输出一个文件内容。浏览器访问本程序的结果如图 10.5 所示。

header()函数除了上述三种常见的应用外，还有一些其他应用，以下是使用 header()函数的一些例子。

【例 10.6】 强制用户每次都从网站获取页面的最新资料，而不是 Proxy（代理服务器）或者浏览器的缓存中的资料，可以使用下列 header()函数之一。

```
<?php
//告诉浏览器此页面的过期时间(格林尼治时间),只要是以前的日期即可
header("Expires: Mon, 26 Jul 1970 05:00:00 GMT");
//告诉浏览器此页面的最后更新日期(格林尼治时间)是当天,目的就是强迫浏览器获取最新资料
header("Last-Modified: ".gmdate("D, d M Y H:i:s")." GMT");
//告诉客户端不使用浏览器的缓存
header("Cache-Control: no-cache, must-revalidate");
header("Pragma: no-cache");      //适用于 HTTP 1.0 协议
?>
```



图 10.5　访问 ex10_5.php 的结果

【例 10.7】 输出状态码 401 到浏览器，要求用户输入用户名和口令，它主要用于用户认证，具体内容见 10.2 节。

```
<?php
header('HTTP/1.1 401 Unauthorized');
header('status: 401 Unauthorized');
?>
```

【例 10.8】 在浏览器中显示“找不到网页”信息。要限制某类用户不能访问该网页，则可将响应的状态码设置为 404，这样浏览器就显示“找不到网页”。

```
<?php
header('HTTP/1.1 404 Not Found');
header("Status: 404 Not Found");
?>
```

需要注意的是，header()函数必须在发送任何输出（包括 HTML 标记、空行）之前执行。使用 header()函数时，最容易引起错误的有两种情况：一是用 include()或 require()函数或者其他文件存取函数来读取程序代码；二是在执行 header()函数之前输出了空格或者空行。

例如，访问以下 PHP 程序时会显示一个错误信息“Warning: Cannot modify header information - headers already sent by”。因为在 header()之前输出了 HTML 的<html>标记。

```
<html>
<?php
header("Location: http://www.php.net/");
?>
```

10.2 用户认证

用户认证（authentication）是一种验证用户身份的安全性机制。用户可以用多种方式证明其身份，最常见的是通过用户名和密码来证明。很多 Web 网站都提供了公共区域和私人区域，公共区域的内容可以被匿名访问，而私人区域的内容则要求用户首先输入自己的身份凭证（用户名和密码）进行验证，验证通过后，才能给用户授予访问 Web 服务器上的资源的权限。这种根据不同权限决定可否得到请求的资源的过程称为授权（authorization）。只有通过授权检查后，系统才允许用户查看页面，调用服务器端脚本程序执行各种任务。用户身份认证方法主要分为 HTTP 基本认证和摘要认证。

HTTP 基本认证是多数主流的浏览器提供访问控制的标准方法。多数 Web 服务器在 HTTP 协议层支持基本认证。HTTP 基本认证根据其实现方法的不同，可分为基于 Apache 服务器的基本认证、基于 PHP 的基本认证、基于数据库的基本认证和基于 IP 地址的基本认证。

10.2.1 HTTP 基本认证的原理

HTTP 基本认证是 HTTP 协议之上的一种相当简单的、有效的用户认证。它由 Web 服务器程序验证用户提供的用户身份凭证是否被接受，如果用户凭证被接受，那么用户就可以访问受保护的任何页面。HTTP 基本认证的过程如下：

（1）客户端浏览器向服务器请求一个受保护的网页。它通过 HTTP GET 方法发送请求，HTTP 请求类似如下内容。

```
GET /members/admin.php HTTP/1.1
Host: 172.16.19.53
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)
Accept: */*
Accept-Language: zh-cn
Connection: keep-Alive
```

（2）服务器收到客户请求的资源是一个受保护的网页，于是向浏览器发送特殊的“401 Unauthorized”响应消息，作为该请求的应答。响应信息如下：

```
HTTP/1.1 401 Authorization Required
Date: Thu, 17 Aug 2006 02:35:29 GMT
Server: Apache/2.0.58 (Win32) PHP/5.1.4
WWW-Authenticate: Basic realm="会员区"
Content-Length: 488
Keep-Alive: timeout=15, max=100
```

```
Connection: Keep-Alive
Content-Type: text/html
```

其中，WWW-Authenticate 之后的关键字 Basic 为基本认证方式，还有一种摘要认证方式（Digest），realm 域的值为区域名称，显示在浏览器的用户认证输入对话框中。Basic 认证方式使用的密码未经加密，直接在网络中传输；Digest 认证方式使用加密的密码，这种方式更加安全。

（3）客户端浏览器收到 401 应答后，显示一个与图 10.6 类似的认证对话框，要求用户输入服务器所需要的用户身份凭证。

（4）用户提供了身份凭证后，再次通过 GET 方法向服务器发送 HTTP 请求，内容如下。

```
GET /members/admin.php HTTP/1.1
Accept: */*
Accept-Language: zh-cn
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)
Host: 172.16.19.53
Connection: Keep-Alive
Authorization: Basic dXNlcjE6MTIzNDU=
```

这次发送的 HTTP 请求中增加了一个新的标头 Authorization。此标头表示 HTTP 基本认证信息，其中关键字 Basic 与等号（=）之间的文本是 base64 编码字符串，表示浏览器输入的用户名和密码。利用 PHP 的 base64_decode()函数解析该字符串，得到用户名是 user1，密码是 12345。

（5）服务器收到浏览器发送的第 2 次请求后，解析认证信息，验证用户名和密码。如果验证通过，就向浏览器发送所请求的资源。如果用户提供错误的或者空白的身份凭证，则被拒绝访问，显示如图 10.7 所示的内容。

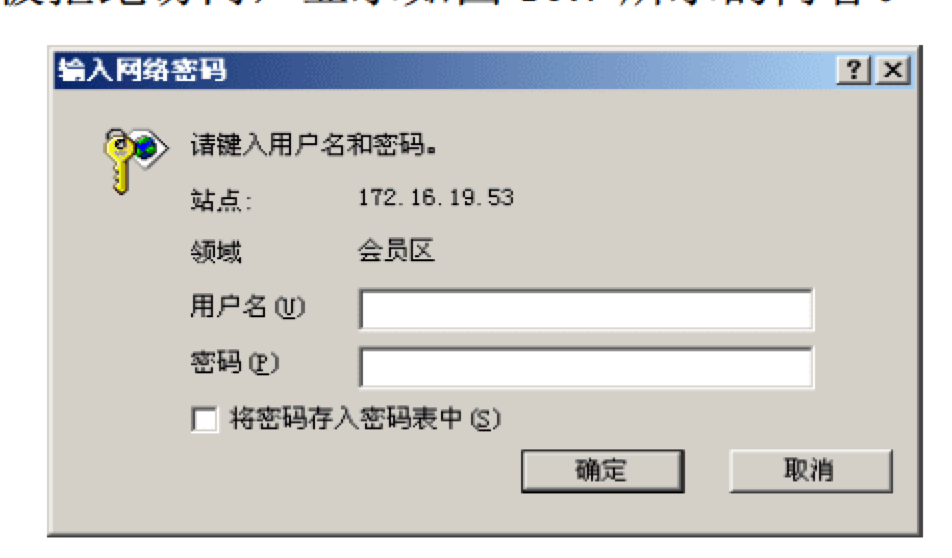


图 10.6　IE 浏览器显示的认证对话框

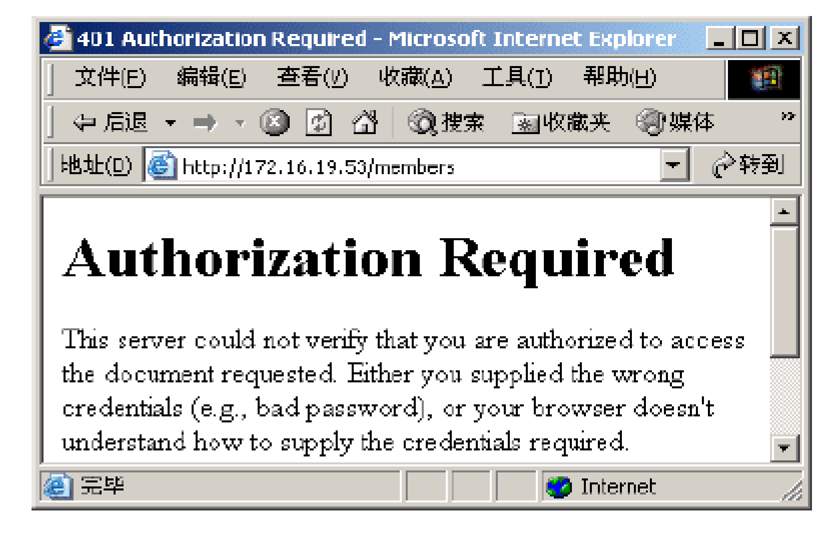


图 10.7　认证失败对话框

需要说明的是，如果用户提供正确的用户身份凭证，那么浏览器在其认证缓冲区中存储认证信息。该缓冲信息一直在浏览器中保存，直到缓冲区被清除，或者向浏览器发送 401 响应错误信息为止。

HTTP 基本认证存在一个缺点：它不能提供对用户名和密码的加密，因此，这些信息在客户端和服务器之间的网络中传输时，可能会被第三方截取。为了消除危及安全的可能性，通常要求建立一个安全的 HTTP 连接，一般是通过安全套接字层（SSL）来实现。目前，所有主流的 Web 服务器（包括 Apache 和微软 IIS）均支持 SSL 协议。

10.2.2 基于 Apache 服务器的基本认证

用 Apache 实现身份认证的基本原理是对受保护的文件目录建立一个身份认证文件，并在 Apache 服务器的 httpd.conf 配置文件中做一些相应的设置。当访问者请求查看这些资源时，Apache 服务器将检查此身份认证文件，如果这个文件存在，那么服务器将向访问者提出身份验证要求，然后按照这个文件中指定的信息对访问者输入的用户名和口令进行确认，在确认正确的情况下，允许访问者查看相应的服务器资源。

下面以 Linux 的 Apache 2 服务器为例，介绍用 Apache 服务器实现用户的基本认证。假设 Apache 软件安装在/usr/local/apache2 目录中，需要对/usr/local/apache2/htdocs/members 目录的文件进行认证访问，其方法如下所述。

1. 建立口令文件

口令文件存放用于身份认证的用户信息，它是一个文本文件，每行内容为一个用户的用户名和口令，格式为“用户名:口令”。默认时，口令用系统的 crypt()函数进行加密。口令文件可以用 Apache 的 htpasswd 命令来创建。这个命令存放在 Apache 安装目录下的 bin 子目录中。该命令的格式为：

```
htpasswd  -b [-c] 口令文件名 用户名 口令
```

执行此命令，将在口令文件中增加一个新用户或者更改指定用户的口令。其中，参数–b 表示在命令行中给出用户的口令；可选参数–c 表示创建新的口令文件，它在第 1 次添加用户时使用。

例如，执行下面命令，创建口令文件/usr/local/apache2/password.crypt，并增加了两个用户：user1 和 user2，口令分别是 12345、67890。

```
htpasswd -b -c /usr/local/apache2/password.crypt user1 12345
htpasswd -b /usr/local/apache2/password.crypt user2 67890
```

这时，/usr/local/apache2/password.crypt 文件内容形式如下所示。

```
user1:q7ET/htYajpQ2
user2:fZ5GirYPYxj5A
```

应该注意的是，口令文件应该存放在不能被网络访问的位置，以避免被下载。例如，如果/usr/local/apache/htdocs 是 Web 站点根目录，那么不能将口令文件存放到这个目录及其子目录中。

在默认情况下，Linux 操作系统使用 crypt()函数加密用户的口令，Apache 的口令也使用 crypt()函数来加密。如果要创建一个含有几十个用户的口令文件，利用上述方法并不方便，可以以 PHP 的 crypt()函数编写一个脚本程序，实现用户口令的成批加密和输出。

【例 10.9】 编写一个 PHP 程序，成批加密用户的口令，然后输出这些用户名和加密的口令。ex10_9.php 程序代码如下：

```
<?php
  // 提交表单后执行下列代码
```

```
if ( !empty( $_POST['in'] ) ) {
    // 读取文本域内容,存放到数组$inList
    $inContents = $_POST['in'];
    $inList = explode( "\n", $inContents );
     $output="";
    foreach ($inList as $line ) {
      $line = trim( $line );
      // 将字符串变量$line的值分解为两部分,存到两个变量$name、$passwd
      list( $name, $passwd ) = explode( ':', $line );
      $passwd = crypt( $passwd );   //加密口令
      // 把用户名和加密后的口令存放到 $output
      $output .= "$name:$passwd\r\n";
    }
    echo  $output;  // 输出用户名和加密口令
     exit;
}
?>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=gb2312" >
    <title>成批加密用户的口令</title>
  </head>
  <body onLoad="document.getElementById('in').focus()" >
<form action="<?= $_SERVER['SCRIPT_NAME'] ?>" method="post">
  <p>输入用户名和口令。每行的格式为: 用户名:口令 <br />
    <textarea name="in" id="in" rows="6" cols="40"></textarea>
    <br>
    <input  type="submit" value="加密口令">
  </p>
</form>
  </body>
</html>
```

第 1 次访问此脚本程序，显示一个表单，在表单的文本域中输入用户名和口令，每行输入一个用户的用户名和口令，两者之间用冒号分开，如图 10.8 所示。提交表单内容后，第 2 次调用本脚本程序，执行 if 语句，分解来自表单文本域中每行的用户名和口令。然后对每行的口令部分调用 crypt()函数进行加密。crypt()函数使用随机种子数进行加密，因此，相同的明文口令永远不会产生相同的加密口令。加密完所有用户的口令后，将用户名和加密口令输出到浏览器，如图 10.9 所示。这样，就可以很方便地利用复制和粘贴功能将用户名和加密口令保存到上述口令文件中。

应该注意的是，利用此程序生成的用户口令文件，只能在 Linux、UNIX 系统的 Apache 服务器中使用，不能应用于 Windows 平台的 Apache 服务器。因为 Windows 不支持 CRYPT 加密算法。



图 10.8　第 1 次访问时显示的表单

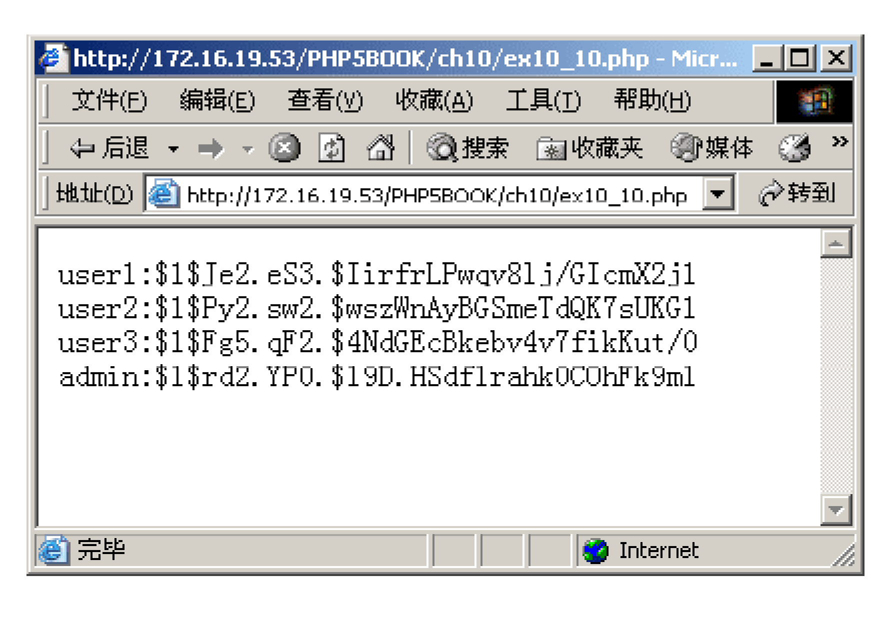


图 10.9　提交表单后输出的加密口令

2．修改 Apache 的配置文件 httpd.conf

在 httpd.conf 配置文件中，对需要保护的目录设置身份认证的相关指令。有两种设置方法。第一种方法是直接在 httpd.conf 文件中对指定目录增加认证指令，例如，对/usr/local/apache2/htdocs/members/目录的访问进行身份验证，在 httpd.conf 文件中增加如下指令。

```
<Directory /usr/local/apache2/htdocs/members/>
    Options Indexes FollowSymLinks
    AuthName "会员区"
    AuthType basic
    AuthUserFile  /usr/local/apache2/password.crypt
    Require valid-user
    Order allow,deny
    Allow from all
</Directory>
```

其中，AuthName 指令指定认证区域名称，可以是任何字符串。区域名称是在浏览器中显示的认证对话框中的领域文字，参见图 10.6。AuthType 指令指定认证方式为基本认证（Basic）。AuthUserFile 指令指定口令文件存放的位置。Require 指令指定哪些用户或组才能被授权访问，其值 valid-user 表示在 AuthUserFile 指令中指定的口令文件中任何用户都可以访问。

采用这种方法时，要求用户具有管理员（root 用户）的权限，才能修改 httpd.conf 文件。如果用户没有修改 httpd.conf 文件的权限，那么需要使用以下第二种方法。

第二种方法是在受保护的目录下创建一个名为.htaccess 的文件，并在该文件中增加一些身份认证指令。例如，在/usr/local/apache2/htdocs/members/目录下创建的.htaccess 文件内容如下：

```
AuthName "会员区"
AuthType basic
AuthUserFile  /usr/local/apache2/password.crypt
require valid-user
```

同时，请求系统管理员修改 Apache 的 httpd.conf 配置文件，增加以下指令。

```
<Directory /usr/local/apache2/htdocs/members/>
    AllowOverride authconfig
</Directory>
```

其中，AllowOverride authconfig 指令表示访问 members 目录下的文件时使用.htaccess 文件的指令进行认证。

由于采用了基本认证方法，每次向服务器请求甚至刷新一个受保护的页面时都必须验证用户名和口令，为此，必须打开口令文件并逐行搜索用户名，从而服务器响应速度受到一些影响，受影响的程度与口令文件的大小成正比。因此，这种基本认证方法适用于用户数量比较少（不超过 50 用户）的情况。如果用户数量有几百个甚至更多时，在口令文件中搜索用户名时需要花掉一定的时间，服务器的响应速度有明显的影响，从而降低了服务器机器的性能。在这种情况下，可以考虑用其他认证方法。

10.2.3 基于 PHP 的基本认证

采用 10.2.2 节 Apache 服务器本身提供的方法进行用户认证，要求系统管理员对 Apache 的 httpd.conf 配置文件作相应的修改。出于安全性和响应速度方面的考虑，系统管理员不允许用户利用.htaccess 文件中定义的指令进行身份认证。在这种情况下，利用 PHP 可以实现同样的身份认证。

使用 PHP 来实现身份认证的原理和 Apache 服务器本身功能提供的身份认证的原理相同。它主要是利用 HTTP 协议的认证方式来实现对用户进行控制。按照前面 10.2.1 节的 HTTP 基本认证原理，在程序的开头通过 PHP 的 header()函数向客户端浏览器发送一个“401 Unauthorized”错误信息，浏览器接收到 401 错误信息后，便会弹出一个用户名 / 密码输入对话框，要求用户输入用户名和密码。当用户输入了用户名和密码并提交给服务器后，PHP 使用全局数组$_SERVER 的三个数组元素来保存这些信息，即$_SERVER["PHP_AUTH_USER"]、$_SERVER["PHP_AUTH_PW"]和$_SERVER["PHP_ AUTH_TYPE"]，这三个数组元素分别存放用户输入的用户名、密码和和认证类型（Basic 或 Digest）。在得到这些信息后，就可以用它们与系统的口令文件或者数据库表的用户信息进行匹配、验证，如果验证成功，则 PHP 程序就发送资源信息到浏览器，否则将报告错误，这些都是由 PHP 程序来控制用户的访问。

为了使用 PHP 进行身份认证，必须将 PHP 设置为 Apache 的模块，才能进行用户认证。如果在 Linux 平台使用 PHP 5 进行身份认证，需要在编译 PHP 时加上--with-apxs2=/usr/local/apache2/bin/apxs 参数，指定 Apache 2 的安装目录（这里假设 Apache 2 安装到/usr/local/apache2 目录）。编译、安装好 PHP 后，修改 Apache 的 httpd.conf 配置文件，增加如下指令，使得 PHP 作为 Apache 的模块。

```
LoadModule php5_module modules/libphp5.so
AddType application/x-httpd-php .php
```

如果在 Windows 平台上使用 PHP 5 和 Apache 2 进行身份认证，需要修改 Apache 的 httpd.conf 配置文件，增加以下指令：

```
LoadModule php5_module "D:/WWW/PHP5/php5apache2.dll"
AddType application/x-httpd-php .php
PHPIniDir "D:/WWW/PHP5"
```

这里，假设 PHP 5 安装在 D:\WWW\PHP5 目录，如果安装在其他目录，则要修改 LoadModule 指令中的路径部分。另外，需要注意的是，在 Windows 下，PHP 程序和 httpd.conf 文件中的路径用斜线（/）将目录名分隔，而不是反斜线（\）。

配置好 Apache 后，下面给出一个通用的 PHP 认证程序，它根据用户输入的用户名和密码与系统存放的口令文件的用户凭证信息进行验证。如果验证成功，继续往下执行程序。如果验证失败，则浏览器继续显示用户名/密码输入对话框。当连续三次输入错误的用户名或密码，则显示“认证失败”，结束程序运行。

【例 10.10】 PHP 认证程序，程序文件名为 authenticate.php。代码如下：

```
<?php
// 文件名 authenticate.php,此脚本必须在另一个脚本中引用
$auth = FALSE;
if (isset($_SERVER['PHP_AUTH_USER']) && isset($_SERVER['PHP_AUTH_PW'])) {
  // 读取口令文件的全部内容,存到一维数组$userList
  $userList=file('passwords.txt');
  // 遍历数组$userList 所有元素
  foreach ($userList as $line) {
    $line = trim($line);
    list($UserName, $Password) = explode(":", $line);
    // 将用户输入的用户名、口令与存储的用户名、口令分别比较
    if ($UserName==$_SERVER['PHP_AUTH_USER'] &&
       $Password==md5($_SERVER['PHP_AUTH_PW'])) {
        $auth = TRUE;
        break;
    }
  }
}
// 判断用户认证是否通过,如果不通过,则浏览器显示输入用户名/密码对话框
if ($auth == FALSE) {
  header('WWW-Authenticate: Basic realm="远程教学区"');
  header('HTTP/1.0 401 Unauthorized');
  echo '认证失败! ';
  exit;
}
?>
```

程序中的口令文件 passwords.txt 可以用例 10.9 的 PHP 程序来生成，但是要将程序中的 crypt 函数名改为 md5，即用 md5()函数来加密密码。passwords.txt 文件与 authenticate.php 文件存放在同一个 Web 目录中。当然也可以保存到其他目录下，只需在 file()函数中指明口令文件的路径。

注意：采用 PHP 实现基本认证时，不管是在 Linux 还是在 Windows 平台，密码的加密只能用 md5()函数来加密，不能用 crypt()函数加密。这一点与 10.2.2 节的 Apache 服务器实现的基本认证有所不同。

在程序中，首先设置认证标志变量$auth 为 FALSE 值。用 isset()函数作为第一个 if 语句的判断条件，即判断用户是否提交了输入的用户名和密码。第 1 次访问此程序时，$_SERVER['PHP_AUTH_USER']和$_SERVER['PHP_AUTH_PW']元素都是空值，因而跳过第一个 if 语句，执行第二个 if 语句，客户端浏览器就显示一个输入用户名和密码的对话框。当用户提交了用户名和密码后，重新调用本程序，此时$_SERVER['PHP_AUTH_USER']和$_SERVER['PHP_AUTH_PW']已被赋值，执行第一个 if 语句，从 password.txt 文件获取每个用户名/密码对清单，存放到一维数组$userList。然后遍历该数组，对数组的每一个元素，用 explode()函数将数组元素的值分解为两部分：用户名、密码，分别用$UserName、$Password 变量来存放。接下来将用户输入的用户名、密码与存储的用户名、密码分别比较，如果比较结果为真值，则将$auth 标志设为 TRUE 值，退出循环，往下执行其他程序。

当用户输入的用户名、密码与口令文件的用户凭证信息不匹配或者输入错误内容时，执行第二个 if 语句。通过执行 header()函数强制向浏览器发送 HTTP 401 错误信息，使浏览器显示一个输入用户名/密码的对话框，参见图 10.6。在 WWW-Authenticate 标头中 realm 域的值是显示在对话框中的领域名称。只要用户输入错误的用户名或密码，都将会引起浏览器显示输入用户名/密码的对话框。此过程至多重复三次，或者取消对话框，退出脚本程序。

此认证程序可以在任何需要认证的程序开头通过 require 语句来引用。下面通过一个例子，说明如何引用此通用认证程序。虽然它并未完成任何实际意义的工作，只是说明其用途而已。

【例 10.11】 引用认证程序 authenticate.php，程序文件名为 ex10_11.php，代码如下：

```
<?php
require 'authenticate.php';
// 如果通过用户认证,则返回到这里,显示一个成功信息
echo '你身份认证成功! ';
// 以下是合法用户可以执行的应用程序部分
?>
```

一般应该将 require 语句写在所有用户都能够访问的脚本（比如主页文件 index.php）的开头。require 语句引用认证程序，如果用户认证失败，则退出。如果用户认证成功，则返回到本程序。在实际的应用软件中，应该在 require 语句后面编写合法用户需要执行的应用程序代码。这样合法用户就可以访问由此程序规定的资源了。

10.2.4　基于数据库的基本认证

在各种基本认证方法中，基于数据库的认证方案是功能最强的方法。因为它不仅增强了管理员的方便性和可伸缩性，而且可以把身份凭证信息集成到数据库中。本节介绍的认证方法是把用户名和密码存储在 MySQL 数据库中，并结合 PHP 程序来实现的。

首先，使用 phpMyAdmin 软件或者 mysql 命令在 MySQL 数据库中创建一个表，用来存储每个用户的用户名和密码等信息。例如，在 student_db 数据库中创建一个名为 userauth 表，它含有 ID、name、username、passwd 四个字段，分别存放记录序号、姓名、用户名和密码，SQL 命令如下：

```
SET NAMES 'gbk';
use student_db;
CREATE TABLE userauth (
    ID TINYINT UNSIGNED NOT NULL AUTO_INCREMENT,
    name VARCHAR(12) NOT NULL,
    username VARCHAR(10) NOT NULL,
    passwd VARCHAR(45) NOT NULL,
    PRIMARY KEY(ID)
) ENGINE=MyISAM DEFAULT CHARSET=gb2312;
```

接下来，给 userauth 表添加以下三条记录：

```
INSERT INTO userauth(name,username,passwd)
        VALUES("李大伟","user1", PASSWORD("12345"));
INSERT INTO userauth(name,username,passwd)
        VALUES("张莉莉","user2", PASSWORD("67890"));
INSERT INTO userauth(name,username,passwd)
        VALUES('李武','user3',PASSWORD('abcd123456'));
```

说明：在添加的记录中，每个用户的密码用数据库函数 PASSWORD()进行加密，以防止别人查看。

下面的例 10.12 给出了根据存储在 userauth 表的用户信息，认证用户输入的用户名和口令的代码。

【例 10.12】 用 MySQL 的 userauth 表中用户凭证信息来认证用户的 PHP 程序，文件名为 ex10_12.php。程序代码如下：

```
<?php
//发送 HTTP 401 代码,显示输入用户名/密码的认证对话框
function authenticate_user() {
    header('WWW-Authenticate: Basic realm="远程教学区"');
    header("HTTP/1.0 401 Unauthorized");
    echo "认证失败,请输入用户名和密码! ";
    exit;
}
//如果用户名是空的,则显示认证对话框,否则在数据库表中查找匹配的记录
if (! isset($_SERVER['PHP_AUTH_USER'])) {
    authenticate_user();
} else {
    mysql_connect("localhost","root","123456") or die("不能连接到服务器!");
    mysql_select_db("student_db") or die("不能打开数据库!");
```

```
        $query = "SELECT name,username, passwd FROM userauth
          WHERE username='$_SERVER[PHP_AUTH_USER]' AND
          passwd=PASSWORD('$_SERVER[PHP_AUTH_PW]')";
        mysql_query("SET NAMES 'gbk'");
        $result = mysql_query($query);
        // 如果找不到匹配的记录,则显示认证对话框
        if (mysql_num_rows($result) == 0) {
            authenticate_user();
        }
        else {    //否则找到记录,表明是合法用户
            echo mysql_result($result,0,"name").", 欢迎你进入远程教学区";
          //在此编写合法用户执行的程序段
        }
    }
    ?>
```

在此程序中，通过调用 authenticate_user()函数，显示输入用户名/密码的对话框。当获得用户输入的用户名后，向 MySQL 发送 SELECT 命令，在 userauth 表查询是否存在符合认证条件的记录。如果找到符合条件的记录（mysql_num_rows()函数值不为 0），则证明该用户是合法的，显示欢迎信息；否则调用 authenticate_user()函数，在浏览器上显示认证对话框。

程序中的 mysql_query("SET NAMES 'gbk'")函数用来设置 MySQL 数据库的字符集为 gbk（简体中文），它适用于 MySQL 4.1 以上版本。该函数必须在第一次执行查询、插入、更新记录之前执行，才能在 MySQL 数据库中存储汉字和显示汉字，否则存储和显示的都是乱码。

10.2.5 基于 IP 地址的基本认证

在某些情况下，需要更高级别的访问控制，不仅要求具备正确的用户名和密码，而且要求限制用户在某些机器上登录，确保用户的合法性。例如，在网络考试系统，需要限制每位考生只能在某个固定 IP 地址的机器上登录。这种情况通常针对一个局域网的站点，其中，某些网页只能被该局域网中的某些机器浏览。

为了达到根据 IP 地址和用户名/密码来认证用户的目的，只需要稍微修改 10.2.4 节的 userauth 表，增加一个 ip_address 字段，用来存放用户可登录的机器 IP 地址。可以执行以下 SQL 命令，增加 ip_address 字段。

```
ALTER TABLE userauth ADD COLUMN ip_address VARCHAR(15) NOT NULL;
```

接下来，对 userauth 表每个记录的 ip_address 字段填上 IP 地址。例如，执行以下 SQL 命令，更新表中记录内容。

```
UPDATE userauth SET ip_address="172.16.19.1" where username="user1";
UPDATE userauth SET ip_address="172.16.19.2" where username="user2";
UPDATE userauth SET ip_address="172.16.19.3" where username="user3";
```

这样，在 userauth 表中存放了每个用户的用户名、密码以及登录的 IP 地址。下面的例 10.13 的程序是根据用户名、密码和 IP 地址进行身份认证的。

【例 10.13】 使用登录信息和 IP 地址进行身份验证。程序文件名为 ex10_13.php，代码如下：

```
<?php
function authenticate_user() {
    header('WWW-Authenticate: Basic realm="网络考试系统"');
    header("HTTP/1.0 401 Unauthorized");
    echo "认证失败,原因是用户名或密码错误,或者未在指定 IP 地址登录!";
    exit;
}
// 主程序
if (! isset($_SERVER['PHP_AUTH_USER'])) {
    authenticate_user();
} else {
    mysql_connect("localhost","root","123456");
    mysql_select_db("student_db") or die("不能打开数据库!");
    $query ="SELECT name,username, passwd FROM userauth ";
    $query.=" WHERE username='$_SERVER[PHP_AUTH_USER]' AND ";
    $query.="      passwd=PASSWORD('$_SERVER[PHP_AUTH_PW]') ";
    $query.=" AND ip_address='$_SERVER[REMOTE_ADDR]'";
    mysql_query("SET NAMES 'gbk'");
    $result = mysql_query($query);
    if (mysql_num_rows($result) == 0) {
        authenticate_user();
    }
    else {
      echo mysql_result($result,0,"name").", 欢迎你进入网络考试系统";
        //在此编写合法用户执行的程序段
    }
}
?>
```

这个程序与上面的例 10.12 的程序大部分相同，只是修改了其中的 SELECT 查询命令的查询条件，利用系统预定义数组元素$_SERVER[REMOTE_ADDR]获取客户机的 IP 地址，用它与 userauth 表存储的 IP 地址进行比较，如果不相同，则表明用户未在指定 IP 地址的机器上登录，拒绝进入系统。

当然，也可以要求用户在指定的 IP 地址段内登录。例如，在 userauth 表中将某用户的 ip_address 字段值改为“172.16.18.*”，允许该用户在网络地址为“172.16.18”的 IP 地址段内登录，此时只需要对上述程序作适当的修改就可以了。如何修改该程序这一问题，留给读者来完成。

注意：通过 IP 欺骗或其他攻击方法可以绕过这种认证方法，因此不要使用这种方法来保护重要信息，比如信用卡号、私人信息等。

上面介绍的四种基本认证方法都有一个缺点：任何人都可以监听在浏览器和 Web 服务器之间网络中传输的未加密的 TCP 数据包，从而有可能获取用户名和密码。因此在未加密的网络上的基本认证只适用于满足最基本安全性的情况，比如社区、网上教学管理等站点，不能用来保护重要信息，如信用卡号或者用户不愿公开的任何其他信息。

10.2.6 Apache 的摘要认证

摘要认证是在 HTTP 1.1 中增加的一个认证方案，它比基本认证更加安全，因为用户的密码不是以明文方式在网络中传递，而是用 MD5 算法将用户密码添加到加密的摘要中，在网络中传递的是摘要信息。利用这种方法，通过嗅探网络信息就无法得到密码了。关于摘要认证的完整说明在 Internet 标准文件 RFC 2617 中可以找到，其网址为 http://www.ietf.org/rfc/rfc2617.txt。

在使用摘要认证方案时，需要浏览器支持摘要认证，目前只有部分浏览器支持它：Amaya、Konqueror、微软 IE 5 以上、Mozilla、Netscape 7、Opera 等，而 lynx 不支持摘要认证。

Apache 的摘要认证由 mod_auth_digest 模块来实现。摘要认证的实现原理与基本认证类似，但认证标头内容更复杂些。其处理过程如下：

（1）浏览器发送要求认证的 URL 请求。

（2）服务器收到请求后，向浏览器发送 401 Unauthorized 错误信息以及 WWW-Authenticate: Digest 认证标头进行响应。摘要认证标头内容比基本认证标头包含更多的信息，如下所示。

```
WWW-Authenticate: Digest realm="Protected Area",
    qop="auth", algorithm="MD5",
    nonce="V8HbMXQbBAA=35ecd547ee438f398aaefaa426898ac92da3e560"
```

实际上，这个标头都是一行的，为了说明和阅读方便，将它分解为三行。Digest 为摘要认证类型。realm 的值与基本认证的含义相同，表示区域名称，它在 httpd.conf 配置文件中定义。每当服务器使用 401 错误代码响应请求时都会生成一个当前值(nonce value)，nonce 值是由客户机 IP 地址、时间戳和只有服务器知道的私有密钥生成的数据，以十六进制格式表示。nonce 值在每次生成时都是唯一的，它是摘要认证的关键。algorithm 的值定义加密信息的算法，其默认值是 MD5，它是单向散列算法，不可能使用计算方法从加密后的信息获得原始信息。qop 指定摘要认证的保证质量，值为 auth，表示认证用户名和密码。

（3）浏览器收到 HTTP 401 错误代码后，显示一个对话框，如图 10.10 所示。输入了用户名和密码后，建立一个授权请求，发送给服务器。该请求是 Authorization 标头形式，如下所示。

```
Authorization: Digest username="user1",realm="Protected Area",
    qop="auth", algorithm="MD5", uri="/private/phpinfo.php",
    nonce="V8HbMXQbBAA=35ecd547ee438f398aaefaa426898ac92da3e560"
    nc=00000001, cnonce="5ace2f84a7029d75b634f65cf052d79c",
    response="4bbb89c7cd7e4c58ab5f150998404da3"
```

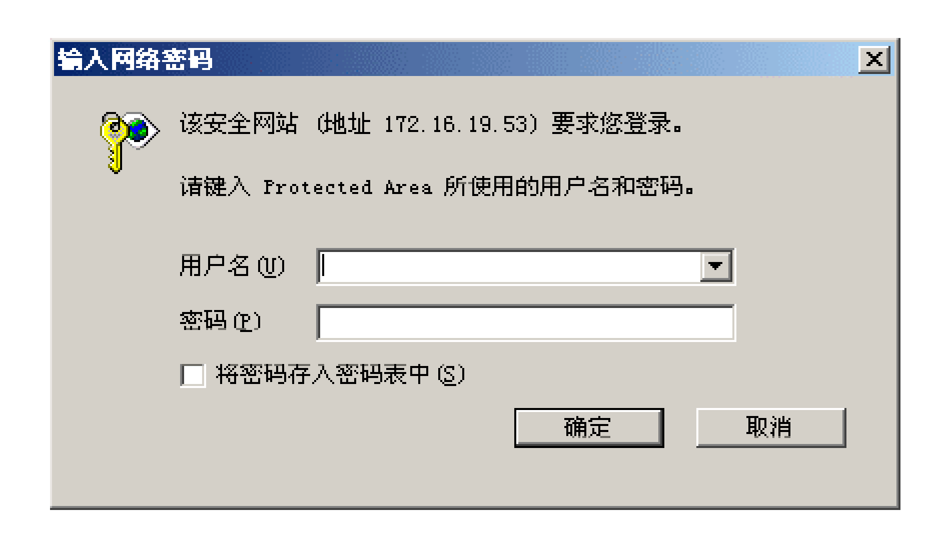


图 10.10　浏览器中显示的摘要认证对话框

这个标头也是一行的，为阅读方便，将其分解为五行。有些字段的值与上面同名字段的含义相同。username 以明文形式存放登录的用户名。nonce 值是服务器所发送的 nonce 值的副本，让服务器知道在响应中使用了哪个 nonce 值。nc 值是 nonce 值被使用次数的计数器。uri 的值是用户请求访问的资源地址。cnonce 值是客户端生成的唯一字符串，可以使客户端在某种程度上控制认证。response 是响应值，称为摘要，它是存放密码的地方，其值是根据用户名、密码、URI、nonce 值、cnonce 值、nc 值、realm 值和 qop 这些信息，使用 MD5 算法进行编码的。

（4）服务器收到这一 Authorization 标头时，使用明文的用户名在口令文件中查找所存储的“用户名:域:摘要散列值”，根据上一次请求中知道 nonce 值（它最先发送）和请求信息。服务器根据这些信息使用相同的算法重新计算摘要，如果由客户发送的摘要与服务器计算得到的摘要相匹配，那么用户就得到认证。

理解了摘要认证的原理后，下面介绍用 Apache 实现摘要认证的具体方法。其实现过程如下。

1．编辑 httpd.conf 文件

在 Apache 2 版本中，利用 mod_auth_digest 模块来实现摘要认证。首先，编辑 httpd.conf 配置文件，在<Directory>、<Location>或者<Files>块放置摘要认证指令。例如，下列指令要求对/private 目录使用摘要认证。

```
Alias /private/ "/usr/apache2/htdocs/private/"
<Directory /usr/apache2/htdocs/private/>
  AuthType digest
  AuthName "Protected Area"
  AuthDigestDomain /private/ http://www.mydomain.com/private/
  AuthDigestFile /usr/local/apache2/passwddig.txt
  Require valid-user
</Directory>
```

上述指令保护/private 目录及其所有子目录。AuthType 指令指定认证类型为摘要认证。AuthName 可以定义区域名称，它显示在用户浏览器的用户名/密码对话框中。AuthDigestDomain 指令定义共享认证信息的多个域，其参数是使用空格分隔的 URI 列表，URI 可以是不同服务器的 Web 位置。因此，客户端可以在这些多个 Web 位置使用相同的用户名和口令。AuthDigestFile 指定口令文件存放的位置，这里是/usr/local/apache2/

passwddig.txt。最后，设置 Require 指令的值为 valid-user，告诉 Apache 在口令文件中的任何用户都可以访问/private 目录下的页面。

在 httpd.conf 文件中，必要时（比如 Windows 版本的 Apache）还需要把以下指令行左边的注释号（#）删除掉，以装载 mod_auth_digest 模块。

```
LoadModule auth_digest_module modules/mod_auth_digest.so
```

2．建立口令文件

使用 Apache 的 htdigest 命令，建立口令文件，格式如下：

```
htdigest [-c] passwdfile realm username
```

可选参数–c 表示创建一个新的口令文件，它在增加第一个用户时使用。passwdfile 参数为口令文件名，realm 为区域名，passwdfile 和 realm 参数的值必须与 httpd.conf 的相应指令的值相同。username 是要创建或更新的用户名。例如，创建口令文件/usr/local/apache2/passwddig.txt，增加第一个用户 user1。

```
# htdigest -c /usr/local/apache2/passwddig.txt "Protected Area" user1
```

按提示输入两次口令。如果在指定的区域中不存在该用户名，则 htdigest 命令增加该用户。如果已经存在该用户名，则修改该用户的口令。

口令文件的格式如下：

```
user1:Protected Area:ffe26dd0bed2c1be0f28f3be8f0829e8
user2:Protected Area:61693eadc81e6d382166d6b1db8714e2
```

口令文件格式的每一行内容的格式为“用户名:区域名:散列摘要值”。散列摘要值不是口令的散列值，而是对“用户名:区域名:口令”字符串用 MD5 算法加密得到的散列值。

一旦完成以上两个步骤后，已经准备好使用摘要认证的环境了，然后重新启动 Apache 服务器。只要用户访问服务器的/private 目录下的任何页面，就会采用摘要认证方式对用户的用户名和密码进行认证。

10.3　PHP 的 Cookie

Cookie 是 Web 服务器通过其页面的程序写到浏览器所在计算机硬盘上的一个数据文件。Cookie 由某一 Web 页面程序创建，并能够被同一个域的其他 Web 页面检索和使用。Cookie 的主要用途是存储用户已经在一个 Web 站点上访问了哪些页面、最后访问时间、访问站点的次数以及用户输入的信息，例如用户名、密码等，这样，当用户下次再访问这个网站时，Web 应用程序就可以检索以前保存在 Cookie 的信息，用户不必每次访问时都输入信息。Cookie 的另一个用途是保存状态管理的信息，利用 Cookie 可以帮助 Web 应用程序保存关于用户的信息，例如，购物网站上的 Web 服务器为了跟踪每个购物者，利用 Cookie 来创建购物车，存放用户选择购买的商品信息。又如，民意调查的网站可以简单地利用 Cookie 的数据，判断用户的浏览器是否已经参加了投票，这在一定程度上避免重复投票。

当用户访问设置了 Cookie 的 Web 站点时，执行 Web 页面的程序，向浏览器发送预先设置的 Cookie 信息，浏览器接收到该信息后将其保存在本地硬盘中的特定位置。不同的浏览器保存 Cookie 的位置各不相同。例如，在 Windows 2000 和 XP 系统中，IE 浏览器把 Cookie 信息保存在类似于 C:\Documents and Settings\LoginName\cookies 文件夹下。当用户再次访问站点时，浏览器就将请求和 Cookie 信息一起发送给服务器。

根据有效期限的不同，Cookie 分为两种类型：临时性 Cookie（亦称会话 Cookie）和永久性 Cookie。临时性 Cookie 将信息保存在用户计算机的内存中。当用户关闭浏览器结束会话过程之后自动清除临时性 Cookie。永久性 Cookie 将信息保存在用户计算机的文本文件中，当用户关闭浏览器后，这些信息仍然保存计算机硬盘中。永久性 Cookie 有一个终止日期，在终止日期之后操作系统将删除该 Cookie。

Cookie 文件的内容包括若干个变量名和相关值对，每个变量名都有一个值，习惯上表示为名/值对，还可以包括有效时间、路径、域、安全参数等。一个单独的 Cookie 文件最多可以保存 20 个名/值对，或者 4096 个字符。如果 Cookie 已经达到了 20 个名/值对的限制，而浏览器又需要在 Cookie 中添加新的名/值对，它将从 Cookie 中删除最先保存的名/值对，再加入新的名/值对。

在网站上使用 Cookie 存在着以下一些限制。

（1）大多数浏览器支持最多可达 4096B 的 Cookie。因此，Cookie 主要用来存储少量的数据，不能用来保存大量数据。

（2）大多数浏览器只允许每个网站保存 20 个 Cookie。如果试图保存更多的 Cookie，则最先保存的 Cookie 将会删除。此外，浏览器还限制 Cookie 总数最多为 300 个。

（3）用户可以设置浏览器，拒绝接受 Cookie。因此，对 Web 开发人员来说，Cookie 不算是一种保存信息的可靠方式。如果 Web 应用程序需要使用 Cookie，则必须先测试浏览器是否接受 Cookie。

10.3.1　创建临时性 Cookie

在 PHP 脚本语言中，可以利用 setcookie()函数来创建和删除 Cookie。setcookie()函数的语法如下：

```
book setcookie(string name [,string value [, int expire [, string path [,
string domain [, bool secure]]]]] )
```

该函数定义一个和其他 HTTP 标头一起发送的 Cookie 信息。Cookie 信息必须在脚本的任何其他输出之前发送，这些输出包括 HTML 标签、空行以及空格。因此必须在任何输出之前调用本函数。如果在调用 setcookie()之前已经输出了内容，本函数将失败并返回 FALSE。如果 setcookie()函数成功运行，将返回 TRUE，但并不说明浏览器接受了 Cookie。

在所有参数中，除了 name 参数以外，其余参数都是可选的。可以用空字符串（""）替换某参数以跳过该参数。参数 expire 和 secure 的值是整型，不能用空字符串，而是用 0 来代替。各参数的含义如表 10.2 所示。

表 10.2　setcookie()函数的参数

参数	说明
name	表示 Cookie 变量名
value	表示 Cookie 变量的值
expire	指定 Cookie 变量值的有效时间值，通常用 time()函数再加上秒数来设定 cookie 的失效期，或者用 mktime()来实现。如果未设定，Cookie 将会在关闭浏览器后失效
path	指定 Cookie 在 Web 服务器端的相应路径，默认值为设定 Cookie 的当前目录
domain	指定 Cookie 的有效域名
secure	指定 Cookie 是否通过安全的 HTTPS 连接传送。当设为 TRUE 时，Cookie 仅在安全的连接中被设置，默认值为 FALSE

要创建临时性 Cookie，只需在 setcookie()函数中给出 name 和 value 这两个参数值即可。下面是一个创建临时性 Cookie 的例子。

【例 10.14】 创建临时性 Cookie 实例，文件名为 ex10_14.php。

```
<?
setcookie("username","王云清");
setcookie("age",20);
echo"已经创建了 cookie.";
?>
```

上述代码中，第 1 条语句创建一个名为 username 的 Cookie 变量，其值为“王云清”；第 2 条语句创建一个名为 age 的 Cookie 变量，值为 20。

当浏览器访问此程序时，就会将程序中的两个 cookie 变量以及它们的值随同页面内容发送到浏览器，并在运行浏览器的计算机内存中存储这两个 cookie 变量的值。

注意：由于 Cookie 是 HTTP 标头的一部分，必须在向浏览器输出 HTML 内容之前执行 setcookie()函数。否则会显示以下错误。除非使用缓冲输出方法，先在服务器保存输出内容，再向浏览器输出。

```
Warning: Cannot modify header information - headers already sent by
```

10.3.2　读取 Cookie

当在客户端设置了 Cookie 后，客户端浏览器会将请求和 Cookie 一起发送回到 Web 服务器。因此，在其他 PHP 程序中便可以通过自动全局数组$_COOKIE 来读取 Cookie 变量的值。如果在 php.ini 配置文件中设置了 register_long_array 指令，也可用$HTTP_COOKIE_VARS 数组来读取 Cookie 变量的值。PHP4.1 以上版本建议使用$_COOKIE 数组。

【例 10.15】 读取 Cookie 信息，文件名为 ex10_15.php。

```
<?
echo "输出 cookie 的信息：<br>";
echo "username 的值：".$_COOKIE["username"]."<br>";
echo "age 的值：".$_COOKIE["age"];
?>
```

当需要读取某一个 Cookie 变量的值时，只需在$_COOKIE 数组的下标中写上 Cookie 变量名，就引用该 Cookie 变量值。比如，$_COOKIE["username"]表示引用 Cookie 中 username 变量的值。

当访问了例 10.14 的程序后，再访问此程序，输出如下三行结果，其中后面两行结果分别由第 2、第 3 条 echo 语句输出。当关闭浏览器后，再打开浏览器直接访问此程序，将不显示这两个 Cookie 变量的值，其原因在于它们是临时性 Cookie。

```
输出 cookie 的信息:
username 的值: 王云清
age 的值: 20
```

通过这个实例的运行结果，可以看出，利用 Cookie 技术，可以使多个 Web 应用程序之间共享 Cookie 信息，达到数据在 Web 程序之间传递的目的。

读取 Cookie 数据，也可以用 foreach 循环语句来实现，如下所示。

```
<?
  foreach ($_COOKIE as $name => $value)
     echo $name."  ".$value."<br>";
?>
```

10.3.3 创建永久性 Cookie

前面已经提过，永久性 Cookie 是存储在用户计算机硬盘上的一个数据文件。为了创建永久性 Cookie，延长 Cookie 的生存期，可以在 setcookie()函数中指定有效时间，这样，关闭浏览器后，Cookie 仍然驻留在用户计算机的硬盘中，以便其后访问同一个 Web 站点时，这些驻留的 Cookie 可以为其他程序所利用。有效时间是一个 UNIX 时间戳（从 1970 年 1 月 1 日零时开始计算的秒数）。可以用两个函数来计算时间戳：第一个是 time()函数，它返回当前的 UNIX 时间戳，对它加上一个秒数来设定 Cookie 的有效时间。第二个是 mktime()函数，它计算一个日期的 UNIX 时间戳。

【例 10.16】 创建永久性 Cookie。

```
  <?php
  $lifetime=mktime(12,30,50,10,8,2009);     //2009 年 10 月 8 日 12:30:50
  setcookie("myuser_id","wang",$lifetime);
  setcookie("test1",1,time()+86400);        //有效期 1 天
  setcookie("test2",2,time()+86400*30);     //有效期 1 个月(30 天)
  //有效期 1 年
setcookie("test3","10",time()+86400*365,"/","www.myweb.com");
?>
```

浏览器访问此程序后，可以在 Windows 2000 和 XP 安装目标盘的\Documents and Settings\LoginName\cookies 文件夹中看到一个新建的 cookie 文件。

【例 10.17】 使用 Cookie 实现的简单计数器。

```
<?
  $count=$_COOKIE["count"]+1;
  setcookie("count",$count,time()+86400*365);
  echo "你访问本页面 $count 次了.";
?>
```

每次访问此程序时，读取 Cookie 变量 count 的值，并增加 1，作为 Cookie 发送浏览器并驻留于硬盘。当第 1 次访问时，该 Cookie 不存在，因此$count 变量初值为 1。

10.3.4 删除 Cookie

删除 Cookie 的方法是重新执行 setcookie()函数，将 Cookie 值设置为空字符串，其余参数和上一次调用 setcookie()函数时相同，这样可以删除客户端指定名称的 Cookie。当然，将有效时间设置为过去任何日期的时间戳，也可以删除指定名称的 Cookie。

【例 10.18】 删除前面例 10.16 中名称为 myuser_id 的 Cookie 变量。

```
<?
setcookie("myuser_id","",time()-365*24*60*60);
?>
```

可以在 Cookie 中使用 Cookie 数组，它的作用相当于多个 Cookie。它的 Cookie 个数和数组中元素的个数相等。在 Cookie 变量名后面加上一对方括号以及下标值，定义 Cookie 数组。下面是使用 Cookie 数组的一个例子。

【例 10.19】 Cookie 数组实例，文件名为 ex10_19.php。

```
<?
//设置 Cookie 数组
if (! isset($_COOKIE['UserInfo']['name'])) {
    setcookie("UserInfo[name]", "黄超");
    setcookie("UserInfo[address]", "广州市");
    setcookie("UserInfo[birthdate]", "1984-8-20");
    header("Location: {$_SERVER['PHP_SELF']}"); //重新导航到本程序
}
else {
//读取 Cookie 数组元素,并输出它们
    $name = $_COOKIE['UserInfo']['name'];
    $addy = $_COOKIE['UserInfo']['address'];
    $bday = $_COOKIE['UserInfo']['birthdate'];
    echo $name."  ".$addy."  ".$bday;
}
?>
```

这个例子首先是判断 Cookie 数组元素 UserInfo['name']是否存在，如果不存在，则创建 Cookie 数组 UserInfo，然后调用 header()函数，向浏览器发送 Location 标头，重新访问本程序，此时，因 UserInfo['name']已被赋值，执行 else 分支，显示 Cookie 数组 UserInfo 的

元素值。

在使用 Cookie 时，容易遇到的问题是并不是所有浏览器都支持 Cookie，或者浏览器禁止了 Cookie 功能。如果程序中使用了 Cookie 而用户的浏览器不支持 Cookie，就需要以一种友好的方式告诉用户，他的浏览器不支持 Cookie。

【例 10.20】 测试浏览器是否支持 Cookie，文件名为 ex10_20.php。

```
<?php
  if (isset($_GET['step']) && $_GET['step'] == '2')
   {
    $test_temp = isset($_COOKIE['test_temp']) ? '支持': '不支持';
    $test_persist = isset($_COOKIE['test_persist']) ? '支持': '不支持';
    setcookie('test_temp', '',time()-365*24*60*60);
    setcookie('test_persist', '', time()-365*24*60*60);
    echo "浏览器 $test_temp 临时性 cookies.<br/>";
    echo "浏览器 $test_persist 永久 cookies.";
  } else {
    setcookie('test_temp', 'ok');
    setcookie('test_persist', 'ok',time()+14*24*60*60);
    header("Location: {$_SERVER['PHP_SELF']}?step=2");
  }
?>
```

这个程序中，首先判断$step 变量是否被赋值，如果未被赋值，则执行 else 分支，设置两个 Cookie 变量，然后通过 header()重新访问本程序，此时因为$step 变量已被赋值，执行 if 分支。如果 Cookie 变量被赋值，则证明浏览器支持 Cookie。

10.4 PHP 的 Session

上一节介绍的 Cookie 技术是在客户端存储用户交互信息的一种手段，由于 Cookie 存在着文件大小最多 4096B、Cookie 数量最多 300 个的限制，而且任何能够使用客户机的人们都可以查看到 Cookie 内容，很不安全。为了解决这些问题，PHP 开发者设计出另一种存储用户交互信息的技术——Session。

10.4.1 Session 的工作原理

从用户角度看，Session 的中文含义是会话，它是从用户登录网站开始，到关闭浏览器或者结束会话所经过的时间。而从 Web 开发者来看，Session 是在服务器端保存的与用户交互相关的变量和信息，其中的变量称为会话变量，记录有关浏览器会话的信息，比如用户名和密码。每个会话可以存储许多变量以及它们的值。当首次启动会话时，服务器生成一个唯一的会话标识符（session ID，SID），它是一个标识会话的长的字符串。通过这个会话标识符，服务器与浏览器保持彼此之间的联系。一旦启动会话后，浏览器每次请求一个页面时，必须把这个会话标识符发送到服务器。服务器根据会话标识符来对用户进行识别，就可以返回用户对应的会话数据。

在默认情况下，会话标识符存放在浏览器的 Cookie 中，这个 Cookie 由 Web 服务器自动发送到访问它的客户端浏览器。也可通过修改配置，在浏览器不支持 Cookie 的情况下使用会话，此时，会话标识符不存入 Cookie，而是手工或自动附加在 URL 后面，作为 URL 参数传递给浏览器并返回。服务器可以通过请求的 URL 获取会话标识符，这在浏览器不支持或禁用 Cookie 功能的情况下非常有用。

根据 PHP 的会话配置不同，可以将会话中的所有信息保存到服务器共享内存、会话文件或者数据库。如果用会话文件来存放会话数据，那么会话标识符作为会话文件名称中的唯一字符串。Web 应用程序根据会话标识符的值，存储或读取用户的会话数据。

会话的生命周期是有限的，默认为 24 分钟，可以通过修改 PHP 的 php.ini 配置文件的指令来设置。在会话的生命周期内，会话是有效的。如果用户的浏览器在超过生命周期的时间里没有向 Web 服务器发送任何请求，那么服务器就会认为这个会话已停止了活动，断开该会话，从而会话就会自动过期，其存储的信息也不再有效。

由于客户端浏览器和服务器之间必须要传输的唯一数据是会话标识符，所有与会话相关的其他数据都驻留在服务器，没有敏感的数据在网络中传输。因此，Session 技术更加安全。

10.4.2　Session 的配置

在 PHP 的 php.ini 配置文件中有一个[session]段，通过设置其中的指令的值，可以调整 Session 的很多处理功能。为了更有效地使用 Session，必须了解这些配置指令，通过 phpinfo() 函数，可以查看服务器当前 PHP 中与 Session 有关的配置，如图 10.11 所示。下面介绍 Session 的部分配置指令，更多的指令参见 php.ini 文件中的说明或者 PHP 帮助指南。

1．session.save_handler

它定义用哪种存储方式来存储会话数据。其值为 files、 mm、sqlite 和 user 之一，默认值为 files。这四种存储方式分别表示采用文件（files）、共享内存（mm）、SQLite 数据库和用户自定义函数（user）来存储会话数据。文件方式可能会产生大量的会话文件。共享内存方式是最快的，也是最不稳定的，因为数据存放在内存中，容易引起系统崩溃。sqlite 方式使用轻量级数据库 SQLite 来透明地管理会话数据。user 方式配置最复杂，但也是最稳定、功能最强大的方式。因为它需要建立自定义处理函数，在任何媒体中存储会话数据。

2．session.save_path

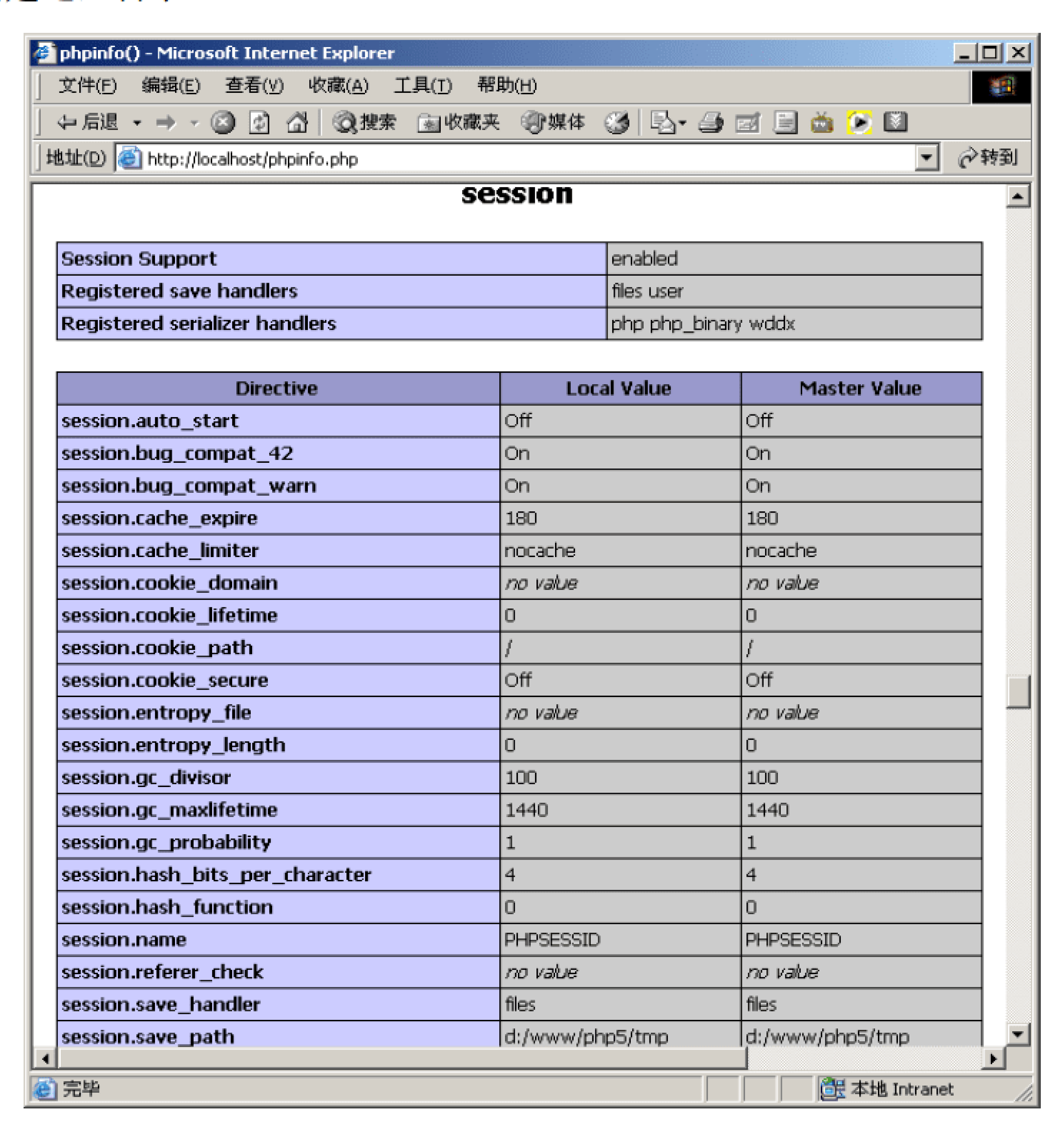
如果 session.save_handler 设置为 files，那么这个指令用来指定存储会话文件的目录。在 UNIX 和 Linux 系统下，其默认值为/tmp，在 Windows 下，必须为此指令设置一个已经存在的目录路径，如 D:/temp，指令如下：

```
session.save_path = D:/temp
```

注意：不能将此参数设置为服务器文档根目录下的任何目录。

其参数值可以定义为“N;/path”的形式，N 为整数，用来指定会话文件所在的目录深度。例如，设定为“5;/tmp”将使创建的会话文件和路径类似于 /tmp/4/b/1/e/3/sess_4b1e384ad74619bd212e236e52a5a174If。要使用 N 参数，必须在使用前先创建好这些目录。如果使用 UNIX 或 Linux 系统，在 ext/session 目录下有个 shell 脚本名叫 mod_files.sh，

可以用来创建这些目录。



Session Support	enabled
Registered save handlers	files user
Registered serializer handlers	php php_binary wddx

Directive	Local Value	Master Value
session.auto_start	Off	Off
session.bug_compat_42	On	On
session.bug_compat_warn	On	On
session.cache_expire	180	180
session.cache_limiter	nocache	nocache
session.cookie_domain	*no value*	*no value*
session.cookie_lifetime	0	0
session.cookie_path	/	/
session.cookie_secure	Off	Off
session.entropy_file	*no value*	*no value*
session.entropy_length	0	0
session.gc_divisor	100	100
session.gc_maxlifetime	1440	1440
session.gc_probability	1	1
session.hash_bits_per_character	4	4
session.hash_function	0	0
session.name	PHPSESSID	PHPSESSID
session.referer_check	*no value*	*no value*
session.save_handler	files	files
session.save_path	d:/www/php5/tmp	d:/www/php5/tmp

图 10.11　phpinfo()函数的输出结果

3．session.use_cookies

指定是否在客户端用 cookie 来存放会话标识符（SID），默认值为 1（启用）。如果设置为 1，则使用 cookie 存放 SID，如果设置为 0，则使用 URL 参数来传递 SID。

注意：当启用 session.use_cookies 后，不需要明确地调用 cookie 的设置函数（如 PHP 的 setcookie()函数），因为这是由 Session 处理程序自动处理的。

4．session.use_only_cookies

指定是否在客户端仅仅使用 cookie 来存放 SID，默认值为 0。设置此指令的值为 1，PHP 只使用 cookie 存放 SID，设置为 0，可以使用 cookie 和 URL 来存放 SID。

5．session.name

指定用来传递 SID 的 cookie 名称。默认值为 PHPSESSID。通过 session_name()函数也可更改其名称。

6．session.auto_start

指定是否自动地启动一个会话，默认值 0（禁止）。建议不要修改此配置，在必要时调用 session_start()函数启动会话。

7．session.cookie_lifetime

指定会话 cookie 的有效期，以秒为单位。默认值为 0，表示会话 cookie 在关闭浏览器之前一直有效。如果要将 cookie 有效期为 1 小时，则本指令应设为 3600，如下：

```
session.cookie_lifetime = 3600
```

8．session.cookie_path

指定会话 cookie 的有效路径，默认值为/。

9．session.cookie_domain

指定会话 cookie 的有效域名，默认值为空串。

10．session.serialize_handler

它定义用来对数据进行序列化和反序列化的回调函数，默认值为 php。当前，PHP 支持内部格式（名为 php）和 WDDX（名为 wddx）两种类型的函数。如果在编译 PHP 时加入一个--enable-wddx 选项，才能使用 WDDX 类型的函数。

11．session.gc_probability

它设置 PHP 会话垃圾（会话文件）收集运行次数的百分比，默认值为 1。如果会话以文件方式来保存，在一定时间内会产生大量的会话文件，因此 PHP 有一个称为“垃圾回收”例程。在每次有会话请求时，PHP 自动调用这个例程来处理会话文件。例如，session.gc_probability 设置为 1，表示在 100 次请求中会有 1 次要清理会话文件。

12．session.entropy_file

指定一个文件来为生成会话标识符的例程提供额外的数据，默认值为空串。例如在许多 UNIX 系统下上，都可以用/dev/random 或/dev/urandom。

13．session.entropy_length (integer)

指定了从上面的外部文件中读取的字节数。默认为 0，表示不能从外部文件读取。

14．session.cache_limiter

指定会话页面所使用的缓冲控制方法。其值为以下五种可能值之一：none、nocache、private、private_no_expire、public。默认为 nocache。

15．session.cache_expire

定义缓冲的会话页面的有效期（秒）。如果 session.cache_limiter 设置为 nocache，则本指令无效。默认为 180。

16．session.use_trans_sid

定义是否自动地在 URL 中传递 SID，默认值为 0（禁用）。如果 session.use_cookies 设置为 0，那么用户唯一的 SID 必须附加在 URL 后面，以便确保 SID 的传送，或者启用本指令，自动地将 SID 附加到 URL 中。

17．session.hash_function

定义用 MD5 算法还是 SHA1 算法来生成 SID。默认值为 0，表示使用 MD5 算法；如果设置为 1，则使用 SHA1 算法。

18．session.hash_bits_per_character

定义组成 SID 的字符数。值为 4、5、6 之一，默认值为 4。设置值为 4，表示 SID 由 0～9、a～f 组成的长度为 32 个字符的字符串；设置值为 5，表示 SID 由 0～9、a～v 组成的长度

为 26 个字符的字符串；设置值为 6，表示 SID 由 0～9、a～z、A～Z、“–” 和 “,” 组成的长度为 22 个字符的字符串。

19．session.gc_maxlifetime

定义会话数据的有效期（秒）。一旦到达此时间限制，会话数据将被销毁，恢复系统资源。默认的会话有效时间为 1440 秒，即 24 分钟。

20．url_rewriter.tags

定义了当 cookie 关闭时哪个 HTML 标记将接收会话标识符。默认值为 a=href，area=href，frame=src，input=src，form=，fieldset=。

10.4.3 Session 的基本使用

1．启动 Session

只有在每个页面上启动会话，才能在会话中创建、修改和删除会话变量，跟踪用户信息。在 PHP 5 中可以在每个 PHP 脚本程序中使用 session_start()函数来启动会话，也可以将 php.ini 文件中的 session.auto_start 指令设为 1，自动地启动会话。session_start()函数的格式如下。

格式：

```
boolean session_start()
```

它根据是否存在会话标识符，创建一个新的会话或者继续当前会话。如果已经存在一个会话标识符（SID），无论 SID 是来自 Cookie 还是 URL，该函数继续当前的会话。如果不存在 SID，则创建一个新的会话。不管 session_start()函数的执行结果如何，该函数的返回值总是 TRUE。

注意：必须在调用 header()函数或者向浏览器输出其他内容之前调用该函数，否则会话不能正常地工作。

2．读写 Session 变量

一旦启动会话，就可以利用$_SESSION 全局数组来给会话变量赋值或者读取会话变量的值。只需在$_SESSION 数组的括号内写上会话变量名作为其下标，就可以存取会话变量了。如果给$_SESSION 全局数组指定一个新的会话变量，那么该会话变量被自动地添加到当前会话中。所有会话变量的值都存放到会话文件或者其他存储媒体中。当给会话变量赋值后，就可以在其后的程序或者其他 PHP 脚本程序中读取会话变量的值。

【例 10.21】 创建会话变量和读取会话变量，程序文件名为 ex10_21.php。

```
<?
    session_start();
    $_SESSION['username'] = "王";
    $_SESSION['age']=20;
    echo "你的名字是".$_SESSION['username']."<br>";
    echo "年龄是 ".$_SESSION['age']."<br>";
    echo "<a href=ex10_22.php>下一个网页</a>"
?>
```

【例 10.22】 读取会话变量，文件名为 ex10_22.php。

```
<?php
    session_start();
    echo "你的名字是".$_SESSION['username']."<br>";
    echo "年龄是 ".$_SESSION['age']."<br>";
?>
```

首先，访问例 10.21 的程序文件 ex10_21.php，输出结果如图 10.12 所示。单击其中的链接，访问 ex10_22.php，输出结果如图 10.13 所示。

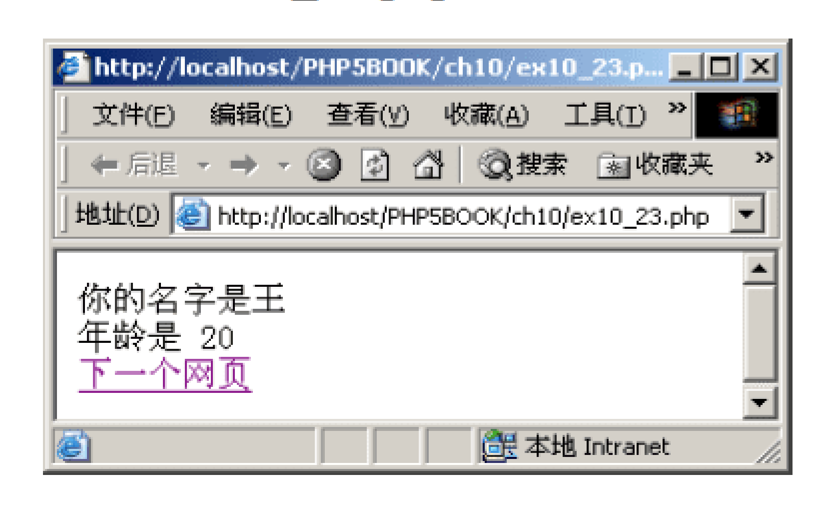


图 10.12　访问 ex10_21.php 的结果

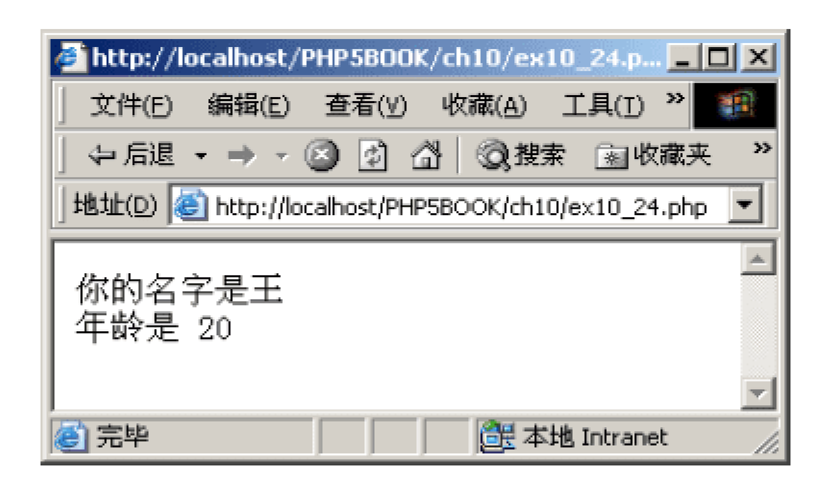


图 10.13　访问 ex10_22.php 的结果

从这两个实例的访问过程可以看出，启动会话后，就可以在其他 PHP 程序中存取一个会话中的所有会话变量。此外，要使用会话变量，必须先调用 session_start()函数，启动会话。

访问了 ex10_21.php 程序后，通过查看存放会话文件的目录，可以看到新增加了一个类似 sess_0924fd353b97ff66a3194666d3d0bf8c 这样的文件。该目录下所有会话文件名都以 sess_开头，其后跟着 32 个字符组成的唯一的会话标识符。该文件的内容如下：

```
username|s:2:"王";age|i:20;
```

会话文件内容由多个会话变量字符串组成，每个字符串又由会话变量名、类型符和值三部分组成，会话变量字符串之间用分号隔开。本例由两个会话变量 username 和 age 组成，username 为字符串型，age 为整数型。会话变量的类型有整数型（i）、浮点型（d）、布尔型（b）、字符串型（s）、数组（a）和对象类型（O）。

3．删除 Session 变量

虽然可以修改 php.ini 配置文件中的[session]段的指令，根据会话的有效时间或者垃圾回收概率，自动地删除会话。但是有时需要在程序中删除会话，例如，用户需要退出网站，单击退出链接，删除全部会话变量。通过调用 session_unset()和 session_destroy()函数，可以删除会话变量。这两个函数如下所示。

1）session_unset()函数

格式：

```
void session_unset()
```

该函数删除当前会话中的所有会话变量，并将会话设置为没有创建会话变量时的状态。

注意：该函数不能完全地删除存储在服务器中的会话，会话文件仍然存在。如果要完全地删除会话，需要使用 session_destroy()函数。

【例 10.23】 用 session_unset()函数删除会话，文件名为 ex10_23.php。

```
<?
   session_start();
   $_SESSION['username'] = '李';
   echo "删除会话前,username 值为".$_SESSION['username']."<br>";
   session_unset();
   echo "删除会话后,username 值为".$_SESSION['username'];
?>
```

访问此程序，浏览器上显示的结果为：

```
删除会话前,username 值为李
删除会话后,username 值为
```

2）session_destroy()函数
格式：

```
boolean session_destroy()
```

该函数删除当前会话中的所有会话变量，并且删除该会话对应的会话文件，使得当前会话无效。

需要注意的是，尽管它删除了会话变量，但是在当前页面上，会话变量仍存在，且变量值也存在。此外，如果用 Cookie 存放会话标识符，它不能删除用户浏览器上的任何 Cookie。如果不想使用会话 Cookie，只需在 php.ini 文件中将 session.cookie_lifetime 指令设置为 0。

【例 10.24】 用 session_destroy()函数删除会话，文件名为 ex10_24.php。

```
<?
    session_start();
    $_SESSION['username'] = 'Michele';
    echo "删除会话前,username 值为".$_SESSION['username']."<br>";
    session_destroy();
    $_SESSION = array();  //删除所有会话变量
    echo "删除会话后,username 值为".$_SESSION['username'];
?>
```

使用 unset()函数，可以删除某个会话变量，其他会话变量仍存在。要注意的是，不能用 unset($_SESSION)来删除整个$_SESSION 数组，否则将不能再通过$_SESSION 全局数组创建会话变量了。

【例 10.25】 删除名为 username 的会话变量（ex10_25.php）。

```
<?php
    session_start();
    $_SESSION['username'] = "Jason";
    $_SESSION['pwd'] = "12345";
    echo "删除前,username 值为".$_SESSION['username']."<br>";
    unset($_SESSION['username']);
    echo "删除后,username 值为".$_SESSION['username'];
?>
```

4. 获取或设置会话标识符

启动会话后，创建了一个会话标识符，它与指定用户的所有会话数据相联系。一般情况下，采用 Cookie 存放会话标识符，此时，PHP 自动地传送会话标识符。但是，如果浏览器关闭了 Cookie，就需要获取会话标识符的值，并在 URL 中发送到浏览器。session_id()函数能够完成这个任务。

格式：

```
string session_id([string sid])
```

该函数可以设置或读取会话标识符，如果未带参数，则返回当前的会话标识符；如果带上 sid 参数，则将会话标识符设置为 sid 的值。例如：

```
<?php
session_start();
echo "会话标识符为".session_id()."<br>";
?>
```

输出结果如下：

```
会话标识符为 66d014deb0ada59d156c51083af1fdaf
```

5. 更改会话标识符

针对受 Session 保护的网站，可能受到的攻击是用户的会话标识符被攻击者盗用，然后攻击者以该用户的角色进入网站，这种情况不容易查出来。为了解决这一问题，可以重新生成一个新的会话标识符，而用户原来的会话数据仍存在，这样攻击者就很难得到用户的会话标识符，因而进不了网站。例如，Amazon 网站要求已经被认证的用户在订购商品时重新登录。更新会话标识符的函数如下。

格式：

```
bool session_regenerate_id([bool delete_old_session])
```

该函数生成一个新的会话标识符，并替换当前的会话标识符，而当前会话的所有会话数据仍保留下来。可选参数 delete_old_session 表示是否删除当前会话标识符对应的会话文件，默认值为 FALSE。

【例 10.26】 更改当前会话标识符（ex10_26.php）。

```
<?php
  ob_start();
  session_start();
  echo 'Old SID: ' . session_id();
  session_regenerate_id();
  echo '<br>New SID: ' . session_id();
  ob_end_flush();
?>
```

输出的可能结果如下：

```
Old SID: e6f7e14f7c310ce6b7cea19efac64c86
New SID: 19f76191f98ae06ee4928cc64c87043d
```

说明：本程序使用 ob_start()和 ob_end_flush()实现输出缓冲。因为 session_regenerate_id()必须在发送 HTML 文档之前被调用。

6．获取或设置 Session 名称

函数格式：

```
string session_name([string name])
```

该函数获取或者设置当前会话名称。如果未带参数 name，则返回当前会话名称。如果带上 name 参数，则将当前会话名称更改为 name 的值。默认的会话名称为 PHPSESSID，在 php.ini 文件指定。

使用 session_name()和 session_id()函数，可以创建动态的含有会话标识符的超链接。这种方法应用于浏览器关闭了 Cookie 的场合。例如，以下 PHP 程序创建的超链接 URL 中将会话标识符作为 URL 参数，传递给 Web 服务器。

```
<?
$name = urlencode(session_name());
$id = urlencode(session_id());
echo "<a href=\"page.php?$name=$id\">访问 page.php</a>";
?>
```

上述代码创建一个含有会话标识符的动态的超链接，使程序员能够创建具有会话的动态链接。这个代码的一个可能输出如下：

```
<a href="page.php?PHPSESSID=5b25b64843fad0d1afb4286cfe0ed448">访问 page.
php </a>
```

7．编码和解码 Session 数据

不管采用哪种方式来存储会话数据，PHP 都将会话数据按一个会话变量由一个字符串组成的标准格式来存储。例如，由 username 和 loggedon 这两个会话变量组成的会话数据内容如下：

```
username|s:5:"jason";loggedon|s:19:"2006-09-01 20:42:48";
```

会话变量条目之间用分号隔开，每个会话变量条目由名称、类型和值三部分组成，形式为：

```
name|s:length:"value";
```

一般情况下，PHP 自动地对会话变量及其值进行编码，自动地对会话变量条目字符串进行解码。但是，有时需要在程序中执行这些任务，就需要使用 session_encode()和 session_decode()函数来完成这些任务。

1）session_encode()函数

格式：

```
string session_encode()
```

该函数将当前会话中的所有会话变量编码为符合标准格式的字符串。返回值为一个字符串，该字符串包含有被编码的当前会话数据。

在应用中，可以将编码后的这个字符串插入到数据库，然后从数据库中检索出来，最后用 session_decode()函数进行解码，得到当前会话的会话变量。例如，在 MySQL 创建一个名为 user 的数据库，在 user 数据库中创建一个 usersession 表，用来存放每个登录到网站的用户会话数据字符串。SQL 命令如下：

```
CREATE DATABASE user;
USE user;
CREATE TABLE usersession(
  id INT NOT NULL AUTO_INCREMENT,
  sid CHAR( 32 ) NOT NULL,
  data TEXT,
  PRIMARY KEY(id)
) TYPE = MYISAM CHARACTER SET gbk COLLATE gbk_chinese_ci;
```

【例 10.27】 用 session_encode()对会话数据进行编码，并存放到 MySQL 数据库中（ex10_27.php）。

```
<?php
session_start();
// 设置会话变量,它们的值可以来自于 HTML 表单
$_SESSION['username'] = "jason";
$_SESSION['loggedon'] = date("Y-m-d H:i:s");
// 将所有会话数据编码为一个字符串,并返回字符串
$sessionVars = session_encode();
echo $sessionVars;
$sid = session_id();
mysql_connect("localhost","root","123456");
mysql_select_db("user");
$query = "INSERT INTO usersession(sid,data)
    VALUES('$sid','$sessionVars')";
$result = mysql_query($query);
echo "<br>".session_id();
mysql_close();
?>
```

返回结果如下：

```
username|s:5:"jason";loggedon|s:19:"2006-09-01 20:42:48";
```

说明： session_encode()函数将该用户的所有会话变量进行编码，而不仅仅对 session_encode()所在的脚本中的会话变量进行编码。

2）session_decode()函数

格式：

```
boolean session_decode(string session_data)
```

该函数对指定的会话数据进行解码，并设置会话变量。session_data 参数表示编码的会话数

据字符串。该函数执行成功时，返回 TRUE；否则，返回 FALSE。

【例 10.28】 假定在 MySQL 数据库中存放了若干条用户会话数据，从 usersession 表中读取与当前会话标识符相同的 data 列数据，进行解码（ex10_28.php）。

```
<?php
session_start();
$sid = session_id();
mysql_connect("localhost","root","123456");
mysql_select_db("user");
$query = "SELECT data FROM usersession WHERE sid='$sid'";
$result = mysql_query($query);
$sessionVars = mysql_result($result,0,"data");
session_decode($sessionVars);
echo "用户 ".$_SESSION['username'].",登录时间 ".$_SESSION['loggedon'];
?>
```

返回的结果为：

```
用户 jason,登录时间 2006-09-01 20:42:48
```

说明：这种方法不是在非标准存储媒体中存储会话数据的首选方法。可以自定义 Session 处理函数，将这些处理函数直接与 PHP API 结合，实现 session 数据的定制处理。具体方法见后面的 10.4.4 节的内容。

熟悉了会话处理的基本函数后，下面介绍一个会话的简单应用例子——用会话实现用户认证信息的跟踪。一般情况下，一旦用户登录后，通过提供唯一标识用户身份的用户名和口令，用户可以很方便地返回到网站而不需要重新登录。其方法很简单：用户通过认证后，将认证信息存储到会话变量，然后在所有受保护的页面的开头，检查该会话变量是否存在。如果该会话变量存在，用户可以透明地访问网站，不需要登录。如果该会话变量不存在，则显示登录表单，重新登录。这里利用会话和 MySQL 表来实现这一方法。以 10.2.4 节的 student_db 数据库的 userauth 表存储用户的账号和口令。首先创建一个含有登录表单的网页文件，如例 10.29 所示。

【例 10.29】 创建一个包含登录表单的网页文件（login.html）。

```
<form method="post" action="<?php echo $_SERVER['PHP_SELF']; ?>">
  用户名：<input type="text" name="username" size="10" /><br />
  口令：<input type="password" name="pswd" SIZE="10" /><br />
  <input type="submit" value="登录" />
</form>
```

此网页文件供下面的例 10.30 的程序引用。

【例 10.30】 使用会话跟踪用户登录信息（ex10_30.php）。

```
<?php
session_start();
// 是否初始化过会话
if (! isset($_SESSION['name'])) {
```

```
        // 如果没有初始化过会话,提交了表单吗
        if (isset($_POST['username'])) {
            $username = $_POST['username'];
            $pswd = $_POST['pswd'];
            //连接 MySQL 服务器,打开数据库
            mysql_connect("localhost","root","123456");
            mysql_select_db("student_db");
            //在 users 表中查找用户
            $query="SELECT name,username FROM  userauth ";
            $query.="WHERE username='$username' AND passwd=PASSWORD('$pswd')";
            mysql_query("SET NAMES 'gbk'");
            $result = mysql_query($query);
            //如果找到用户,设置会话变量
            if (mysql_num_rows($result) == 1) {
                $_SESSION['name']=mysql_result($result,0,"name");
                $_SESSION['username']=mysql_result($result,0,"username");
                echo $_SESSION['name'].",你成功登录了。<br>";
            } else {
                include "login.html";
                exit;
            }
        // 如果未提交表单,显示登录表单
        } else {
            include "login.html";
            exit;
        }
    //已经初始化过会话
    } else {
        $name = $_SESSION['name'];
        echo "欢迎回来, $name! <br>";
    }
    ?>
    <a href="logout.php">退出</a>
```

该程序先判断会话变量 name 是否存在，如果 name 会话变量不存在，说明用户还没有登录，因此再进一步判断用户是否提交了登录信息。如果提交了登录信息，即存在 $_POST['username']变量，那么，根据输入的用户名和口令信息，在 userauth 表中查找匹配该用户的记录，找到后，创建 name 和 username 两个会话变量，分别存放姓名和用户名。如果找不到匹配的记录，则使用 include 语句来引用 login.html 文件，显示表单，重新登录。在会话有效期内，用户关闭了浏览器，又打开浏览器，重新访问受保护的页面程序，此时，因 name 会话变量已经存在，不需要再次登录了。

在需要受保护的所有页面的开头，利用 require("ex10_30.php")语句引用本程序，就可以达到用会话来跟踪用户认证信息的目的。

10.4.4 创建定制的 Session 处理程序

在默认情况下，会话数据以会话文件形式存放在服务器中。当访问者数量有几百、几千甚至上万人时，这种方式会产生成千上万个会话文件，这种在文件中存储会话方式存在

一些缺陷：一是造成访问文件的速度下降了；二是安全性问题。因为会话文件是以明码方式存储的，容易被人查看其内容。存储会话数据的最好方法是采用数据库，因为采用数据库存储会话数据，开销最小，具有数据安全性控制能力，并且可以在任何运行 PHP 的平台上有效。

用户自定义的 Session 处理程序是四种会话存储方式中最灵活、最稳定的方法。为了实现定制的 Session 处理程序，不管选用哪种数据库或者其他形式的文件来存储会话，必须遵循以下两个规则。

首先，开发人员必须定义以下六个会话处理函数，每个函数满足 PHP 的一个会话处理功能。另外，不管函数体是否使用参数，在函数定义中必须有相应的参数定义。这六个函数的结构和功能如下所述。

（1）session_open($session_save_path, $session_name)：初始化在会话过程使用的资源，例如，打开用于存储会话的文件或者数据库。$session_save_path 和$session_name 参数分别表示会话文件的路径和会话名称，这两个参数的用处不大。

（2）session_close()：关闭 session_open()初始化的资源，它的用处不大。记住，该函数不删除会话，删除会话由 session_destroy()函数来完成。

（3）session_read($sessionID)：根据会话标识符读取会话数据，$sessionID 为会话标识符。

（4）session_write($sessionID, $value)：将编码的会话数据写入存储媒体（如数据库或文件）中。$sessionID 是会话标识符，$value 是会话数据。

（5）session_destroy($sessionID)：删除会话及其所有会话变量，$sessionID 为当前的会话标识符。

（6）session_gc($lifetime)：清理会话垃圾，删除所有已经超过会话生命周期的会话。$lifetime 为会话生命周期，其值为 php.ini 文件中 session.gc_maxlifetime 指令的值。

说明：上述六个函数的名称可以由用户自由地指定，并非一定是这些函数名称。

其次，利用 PHP 的 session_module_name()函数将会话存储方式设置为“user”，设置会话存储为用户方式，这样就可以由程序员自定义会话存储的媒体，如数据库、文本文件等。此外，还必须将上述六个自定义会话处理函数与 PHP 的会话处理逻辑进行整合，方法是通过给 session_set_save_handler()函数传递这些函数的名称来实现。注意，必须按照打开会话、关闭会话、读取会话、写入会话、删除会话和清理会话垃圾的顺序给 session_set_save_handler()函数传递这些函数名称，代码如下：

```
session_module_name("user");
session_set_save_handler("session_open","session_close",
    "session_read","session_write", "session_destroy","session_gc");
```

理解了自定义 Session 处理的规则后，下面以 MySQL 数据库存储会话为例，说明定制的 Session 处理程序的实现过程。

在 MySQL 数据库系统中创建一个名为 sess_db 的数据库，然后在该数据库中创建一个存储会话数据的 sessioninfo 表，它包含四个字段：SID 字段存放会话标识符；name 字段存放登录的用户名，为表的主键；expiration 字段存放会话的有效时间（秒）；value 字段存放

编码的会话数据。SQL 命令如下：

```
CREATE DAATBASE sess_db;
USE sess_db;
CREATE TABLE sessioninfo (
    SID CHAR(32) NOT NULL,
    name varchar(20) NOT NULL,
    expiration INT NOT NULL,
    value TEXT NOT NULL,
    PRIMARY KEY(SID)
);
```

例 10.31 的程序是按照上述定制的 Session 处理规则来编写的自定义会话处理程序。

【例 10.31】 用 MySQL 数据库存储会话数据的自定义 Session 处理程序，文件名为 mysqlsessionhandler.php。程序代码如下：

```
<?php
//与 MySQL 永久连接,打开 sess_db 数据库
function mysql_session_open($session_path, $session_name) {
    mysql_pconnect("localhost", "root", "123456");
    mysql_select_db("sess_db") or die("不能打开 sess_db 数据库");
} // end mysql_session_open()
//此函数未执行任何操作,因为服务器是永久性连接的。但必须定义它
function mysql_session_close() {
    return 1;
} // end mysql_session_close()
//从数据库读取会话数据
function mysql_session_read($SID) {
    $query = "SELECT value FROM sessioninfo WHERE SID = '$SID'
      AND expiration > ". time();
    $result = mysql_query($query);
    if (mysql_num_rows($result)) {
        $row=mysql_fetch_assoc($result);
        $value = $row['value'];
        return $value;
    } else {
        return "";
    }
} // end mysql_session_read()
//将会话数据写入数据库,如果 SID 已经存在,则更新现有的会话数据
function mysql_session_write($SID, $value) {
    $lifetime = get_cfg_var("session.gc_maxlifetime");
    $expiration = time() + $lifetime;
    $name=$_POST['username'];
    $query = "INSERT INTO sessioninfo VALUES";
    $query.= " ('$SID','$name', '$expiration', '$value')";
```

```
        $result = mysql_query($query);
        if (! $result) {
            $query = "UPDATE sessioninfo SET expiration = '$expiration',";
            $query.="value = '$value',name='$name' WHERE ";
            $query.=" SID = '$SID' AND expiration >". time();
            $result = mysql_query($query);
        }
    } // end mysql_session_write()

    //根据输入的 SID,删除对应的一行会话数据
    function mysql_session_destroy($SID) {
        $query = "DELETE FROM sessioninfo WHERE SID = '$SID'";
        $result = mysql_query($query);
    } // end mysql_session_destroy()

    //删除所有已经过期的会话,$lifetime 参数为会话的生命周期(秒)
    function mysql_session_gc($lifetime) {
       $expiration=time() - $lifetime;
        $query="DELETE FROM sessioninfo WHERE expiration<".$expiration;
        $result = mysql_query($query);
        return mysql_affected_rows($result);
    } // end mysql_session_gc()

    session_module_name("user");
    session_set_save_handler("mysql_session_open", "mysql_session_close",
        "mysql_session_read",
        "mysql_session_write",
        "mysql_session_destroy",
        "mysql_session_gc");
    ?>
```

由于采用 MySQL 数据库存储会话，因此，将连接 MySQL、打开数据库的语句写在 mysql_session_open()函数中。在运行 session_start()之后，PHP 立即运行 mysql_session_open()函数，然后运行 mysql_session_read()来读取当前会话数据（如果会话记录存在）。

mysql_session_close()是在一个会话读或写操作完成时触发执行的。该函数在此不起多大作用，因为 MySQL 是永久连接的，不需要关闭其连接。但是，必须定义该函数。

mysql_session_read()根据会话标识符返回包含该会话数据的字符串，如果无会话数据，则返回一个空字符串。其中，mysql_fetch_assoc()从查询结果集中返回一行作为关联数组，编码的会话数据在数组元素$row['value']中。

mysql_session_write()将会话数据写入数据库。如果会话记录不存在，则插入会话记录；否则更新已经存在的会话记录。为了便于统计，在 name 字段存放登录用户名。get_cfg_var()读取 php.ini 配置文件中 session.gc_maxlifetime 指令的值，得到会话的生命周期。当创建会

话变量时触发 mysql_session_write()函数的执行。

mysql_session_destroy()删除当前的会话记录。当执行 PHP 的 session_destroy()时触发该函数的执行，从而删除 sessioninfo 表中当前的会话记录。它用于在退出时删除会话。

mysql_session_gc()删除已经过期的会话记录。对于因用户关闭浏览器而没有删除会话时，会话记录仍保留在数据表中，PHP 就调用此函数来删除过期的会话。过期时间的计算是当前时间减去 PHP 提供给 session_gc()的会话生命周期。

session_set_save_handler()告诉 PHP 哪个用户编写的函数将执行哪个会话处理功能。该函数和 session_module_name()函数必须在 session_start()之前执行。否则自定义会话处理程序不能工作。

当需要用 MySQL 数据库存储会话数据时，必须在程序开头用 include "mysqlsessionhandler.php"语句引用上述程序，然后启动会话。为了测试自定义 Session 处理程序的运行过程，使用以下例 10.32 的脚本程序来验证。

【例 10.32】 测试自定义 Session 处理程序的运行（test_mysqlsession.php）。

```
<?php
include "mysqlsessionhandler.php";
session_start();
if (! isset($_SESSION['name'])) {
   if (! isset($_POST['username'])) {
    INCLUDE "login.html";
     exit;
   } else{
     $_SESSION['name'] = $_POST['username'];
     $_SESSION['passwd']=$_POST['pswd'];
   }
}
echo $_SESSION['name'].",欢迎进入本网站<br>";
echo "<a href='logout2.php'>退出</a>";
?>
```

访问此程序，输入用户名 user1，口令 12345。登录后，查看 sessioninfo 表内容，增加了一条会话记录，内容如下。

SID	name	expiration	value
d1b0268dda106f8e1166f1cfbef6735c	user1	1157466793	name|s:5:"user1";passwd|s:5:"12345";

logout2.php 脚本程序内容如下，它用来删除当前的会话记录。

```
<?
INCLUDE "mysqlsessionhandler.php";
session_start();
session_destroy();
$_SESSION=array();
?>
```

10.4.5 显示在线用户

当使用 10.4.4 节介绍的用 MySQL 存储会话数据这一解决方案时，可以很容易地列出在线的用户名，统计在会话生命周期内在线的用户数、未登录的访客数量等。下面是这一问题的实现代码（文件名 online_rs.php）。其中，$count 变量统计在线人数，$others 统计访客的人数。

```
<?
mysql_pconnect("localhost", "root", "123456") or die("不能连接 MySQL 服务器!");
mysql_select_db("sess_db") or die("不能打开 sess_db 数据库");
$lifetime = get_cfg_var("session.gc_maxlifetime");
$expiration =time() - $lifetime;
$sql="select name from sessioninfo where expiration>=".$expiration;
if ($result=mysql_query($sql)) {
    $others=0;
    $count=0;
    while ($row=mysql_fetch_assoc($result)) {
        if (strlen($row['name'])) {
             $onlineuser.= $row['name']."   ";
            $count++;
        }
        else
           $others++;
    }
}
echo "在线用户:".$count."人<br>".$onlineuser;
echo "<br>访客: ".$others."人";
?>
```

实 验 10

1．验证例 10.3 和例 10.4 的显示动态 JPG 图形的程序。

2．设计一个程序，根据存放在文本文件中的计数器数值，输出动态生成的图形计数器。

3．编写一个检测用户的 Cookie 功能是否启动的 PHP 程序。

4．编写一个 PHP 程序，读取一个文本文件内容，将其输出为 Word 文档，并能够显示“文件下载”对话框，以便下载该 Word 文档。

5．更改 Apache 的配置文件 httpd.conf 的内容，实现对 Web 虚拟路径/sale 进行基本认证。假设虚拟路径/sale 对应的物理路径是 C:\sale 文件夹，口令文件存放在 c:\paswddir 文件夹，文件名是 mypassword.txt。

6．验证例 10.10 和例 10.11 结合的 PHP 基本认证程序。

7．设计一个含有表单的网页文件（如图 10.14 所示）和一个 PHP 程序，当在表单的网址文本框中输入一个网址后，单击“提交”按钮，能够在浏览器窗口上显示该网址对应的页面内容。

8．设计一个含有登录表单的网页文件（如图 10.15 所示）和一个 PHP 程序，实现：

访问网页文件，输入用户名和口令，选择相应的用户类别（分为学生、教师和管理员），单击“登录”按钮，能够把用户名、口令和所选的用户类型存储到 Cookie 中。

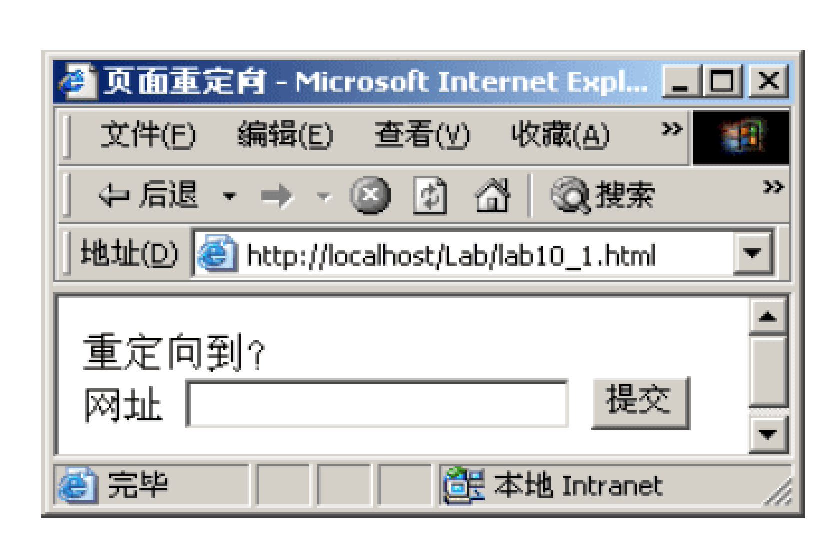



图 10.14　重定向表单



图 10.15　登录表单

9．验证例 10.29 和例 10.30 结合的用会话跟踪用户信息的程序。

10．验证例 10.31 和例 10.32 结合的自定义会话处理程序。

11．编写一个 PHP 程序，创建 yourclass 和 yourname 两个会话变量，分别存放你所在的班级名称和自己的姓名，然后输出这两个会话变量的值。

习　题　10

1．header()函数的功能是什么？举例说明 header()有哪些应用。

2．常见的 MIME 类型有哪些？

3．什么是认证？什么是授权？用户认证有哪几类？

4．如何理解 HTTP 基本认证的原理？

5．HTTP 基本认证的实现可分为哪几种？

6．在 HTTP 基本认证中，PHP 常量$_SERVER['PHP_AUTH_USER']、$_SERVER['PHP_AUTH_PW']分别存放什么值？

7．如何理解 Apache 摘要认证的工作原理？

8．基本认证和摘要认证有何不同？

9．什么是 Cookie？它有哪些主要用途？Cookie 有哪些限制？

10．什么是临时性 Cookie？什么是永久性 Cookie？两者之间有什么差别？

11．利用 setcookie()函数设置永久性 Cookie 时，必须要指定哪个参数？

12．什么是 Session？Session 数据存放在服务器还是客户端计算机？

13．Session 和 Cookie 之间有什么区别？

14．在默认情况下，Session 数据存放哪种存储媒体中？

15．会话文件存放什么内容？其存储格式是怎样的？

16．创建自定义会话处理程序，需要遵循哪些原则？

17．在默认情况下，PHP 的会话的生命周期是多少秒？

第 11 章　XML　基　础

本章导读

XML 是目前各种数据特别是文档的首选格式，在数据交换与各类应用中起着越来越重要的作用。本章主要介绍 XML 的基础知识，包括 XML 的基本概念、主要特点以及 XML 文档的基本结构、语法规则与类型定义。

11.1　XML　概　述

XML 即可扩展标记语言（extensible markup language），与超文本标记语言（hypertext markup language，HTML）一样，均为万维网联盟（World Wide Web Consortium，W3C）所制订的标准。其实，XML 与 HTML 师出同门，都是从标准通用标识语言（standard generalized markup language，SGML）简化而来的。1998 年 2 月，W3C 即发布了 XML 1.0 规范，这也是到目前为止所使用的唯一的一个 XML 标准或版本。

与 HTML 用于创建 HTML 文档类似，XML 则用于创建 XML 文档。XML 文档与 HTML 文档一样，均为纯文本文档，可用各种编辑器（如记事本、写字板等）加以创建。XML 文档的扩展名通常指定为 XML。如例 11.1 所示，即为一个简单的 XML 文档，所描述的是一本图书的有关信息（书名、作者、出版社、出版日期、标准书号）。

【例 11.1】 XML 文档的简单实例（11-1.XML）。

```
<?xml version="1.0" encoding="GB2312" standalone="yes"?>
<book>
 <title>PowerBuilder 数据库应用开发技术</title>
 <author>卢守东</author>
 <publisher>清华大学出版社</publisher>
 <date>2006-06-01</date>
 <isbn>7-302-12729-8</isbn>
</book>
```

可见，XML 文档与 HTML 文档的格式是颇为相似的。用 IE 浏览器（5.0 版以上）打开该 XML 文档（其文件名为 11-1.XML），显示结果如图 11.1 所示。

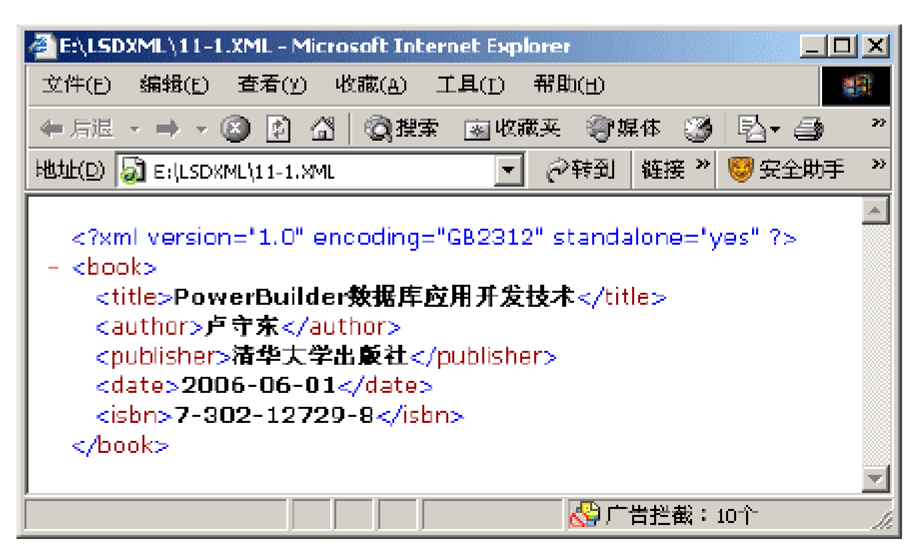


图 11.1　在 IE 中打开 11-1.XML 的结果

与 HTML 相比，XML 主要具有以下特点：

（1）可扩展性。XML 是一种元标记语言，

允许用户根据需要自行创建相应的标记，以便更准确地描述数据。如例 11.1 所示，文档 11-1.XML 中的<book>、<title>、<author>、<publisher>、<date>、<isbn>标记即为用户自定义的标记。相反，HTML 的标记是由 W3C 进行定义与管理的。对于某个版本的 HTML 来说，可供使用的标记是固定不变的。如<html>、<head>、<body>等，即为 HTML 中的常用标记。

（2）数据及其显示相分离。XML 只描述数据，而不表示如何显示。如图 11.1 所示，由于浏览器并不知道如何去处理文档 11-1.XML 中用户自定义的标记，因此只能显示出文档的源代码。相反，HTML 则将数据及其显示结合在一起。对于浏览器来说，由于已经知道如何去处理 HTML 文档中的有关标记，因此可直接显示出相应的结果。例如，HTML 文档中的代码“HTML 与 XML”，在浏览器中将显示为红色的文字“HTML 与 XML”。

目前，XML 已成为各种数据特别是文档的首选格式，在数据交换与各类应用中起着越来越重要的作用。使用 XML，可以轻松地表示各种结构化数据与半结构化数据。此外，使用 XML，也可以制订各种相关标准。如化学标记语言（chemical markup language，CML）、数学标记语言（mathematical markup language，MathML）、无线标记语言（wireless markup language，WML）、同步多媒体集成语言（synchronized multimedia integration language，SMIL）、绘图元语言（drawing meta language，DrawML）、可伸缩矢量图（scalable vector graphic，SVG）、信息和内容交换（information and content exchange，ICE）、商业 XML（commerce XML，cXML）等，均为已使用的基于 XML 的标准。

XML 不依赖于任何操作系统、编程语言或软件供应商，具有极强的平台独立性，易于实现各种跨平台、跨应用程序、跨语言的应用。随着 Internet 的发展及其应用的深入，XML 将在各种分布式应用与系统集成中发挥更重大的作用。

11.2　XML 文档的基本结构

一个完整的 XML 文档通常都包含有声明与主体两个部分。如例 11.1 所示，文档中的第一行即为 XML 文档的声明，其下的各行即为文档的主体部分。

对于一个 XML 文档来说，XML 声明实际上是一个可选项，但 W3C 建议最好使用，以声明相应的文档是一个 XML 文档，并指定该文档所使用的 XML 版本号（version）。还可以指定文档所使用的编码字符集（encoding），以及该文档是否为一个独立的文档（standalone）。在例 11.1 的 XML 声明中，即指定 XML 版本号为 1.0（实际上，1.0 是目前 XML 的唯一版本）、编码字符集为 GB2312（简体中文编码），且该文档为一个独立的 XML 文档。

XML 文档的主体实际上就是一个 XML 元素，通常称之为根元素。当然，可根据需要在根元素中包含其他有关的元素，并称之为根元素的子元素。还可以在子元素中再包含其他子元素。在例 11.1 中，XML 文档的主体即为 book 根元素，其中包含有 title、author、publisher、date、isbn 五个子元素。

11.3　XML 文档的语法规则

为了编写出符合要求的 XML 文档，必须遵循一定的语法规则。XML 具有与 HTML

相类似的基本语法。当然，二者还是有区别的，而且 XML 要比 HTML 要求得更为严格一些。下面分别进行简要的介绍。

11.3.1　标记

作为一种可扩展的标记语言，XML 允许用户自行创建相应的标记，以便更准确地描述有关的数据。

在 XML 中使用标记时，应遵循以下规则：

（1）标记必须以“<”开始、以“>”结束，这与 HTML 中的标记写法是一样的。

（2）标记名必须以字母或下划线（_）开头，后面的字符可以是字母、数字、下划线（_）、短横线（-）或小圆点（.），如<pcbook>、<abc.123>、<_Root>等都是合法的标志，而<pc*book>、<123.abc>、<-Root>等是非法的标记。

（3）标记名中不能包含有空格，如标记<pc book>是不合法的。

在 XML 中，标记必须成对出现，前者称为打开标记（或起始标记），后者称为关闭标记（或结束标记）。作为关闭标记，其名称必须与相应的打开标记的名称保持一致，并在名称前加上一个“/”。例如，对于打开标记<Name>，相应的关闭标记为</Name>。

在 XML 文档中，若使用了非法的标记，或遗漏了相应的关闭标记，则会导致文档无法正常显示。

【例 11.2】 标记使用错误示例（11-2.XML）。

```
<?xml version="1.0" encoding="GB2312" standalone="yes"?>
<book>
 <title>PowerBuilder 数据库应用开发技术</title>
 <author>卢守东
 <publisher>清华大学出版社</publisher>
</book>
```

在该文档中，遗漏了与打开标记<author>相对应的关闭标记</author>，因此是错误的。在 IE 浏览器中打开该 XML 文档(其文件名为 11-2.XML)，显示结果如图 11.2 所示。

应该说明的是，XML 对大小写是敏感的。因此，在使用标记时，应注意其名称的大小写的区别。例如，<Abc>与<abc>就是两个不同的标记。

图 11.2　在 IE 中打开 11-2.XML 的结果

11.3.2　声明

XML 声明用于标明当前文档是一个 XML 文档及其版本信息，指定该文档所使用的编码字符集，以及该文档是否为一个独立的文档。

XML 声明以“<?xml”开始、以“?>”结束，其语法格式为：

```
<?xml version="1.0" [encoding="GB2312|BIG5|…"] [standalone="yes|no"] ?>
```

其中，“[]”中的内容为可选项，“|”则表示“或者”之意。

在 XML 声明中，version 属性指定 XML 的版本号。由于目前只有 XML 1.0 版本，因此 version 属性的值只能设置为“1.0”。version 属性是 XML 声明的必备属性，没有 version 属性的声明是无效的。

【例 11.3】 无效声明示例（11-3.XML）。

```
<?xml encoding="GB2312" standalone="yes"?>
<book>
 <title>PowerBuilder 数据库应用开发技术</title>
 <author>卢守东</author>
</book>
```

在 IE 浏览器中打开该 XML 文档（其文件名为 11-3.XML），显示结果如图 11.3 所示。

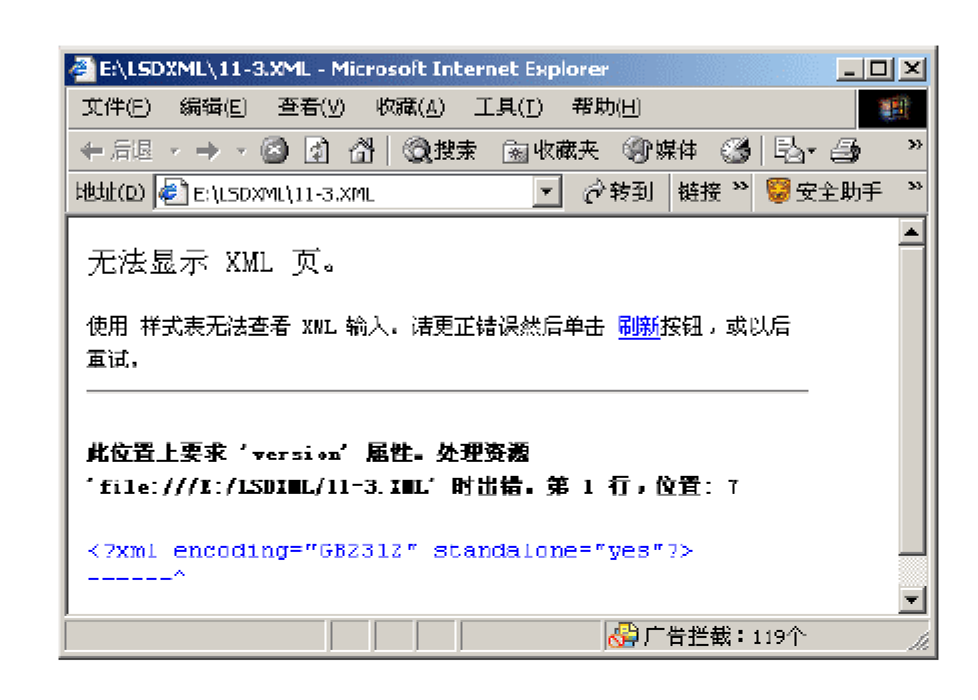


图 11.3　在 IE 中打开 11-3.XML 的结果

Encoding 属性是 XML 声明的可选属性，用于指定当前文档所使用的编码字符集，如 GB2312（简体中文）、BIG5（繁体中文）、GBK（简繁体中文）等。若未指定 encoding 属性，则默认为 Unicode 编码格式。如果所使用的编辑器（如写字板等）可以将文档保存为 Unicode 编码格式，那么在 XML 声明中就无须指定 encoding 属性。当然，为了确保文档的内容能够在浏览器中被正常显示，还是以指定相应的 encoding 属性为好。例如，在文档的内容中包含有简体中文时，通常应将 encoding 属性的值设置为“GB2312”或“GBK”。

Standalone 属性是 XML 的另外一个可选属性，用于指定当前文档是否为一个独立的文档（即是否需要引用其他外部文档）。若其值设为“yes”，则为独立文档（即无须引用其他文档）；若其值设为“no”，则为非独立文档（即可能要引用其他文档）。未指定 standalone 属性时，则默认文档为非独立文档。

实际上，整个 XML 声明也是一个可选项，对于一个 XML 文档来说并不是必需的。但为规范起见，W3C 建议最好使用 XML 声明。在使用 XML 声明时，必须将其置于文档的最前面，即在 XML 声明前不能出现其他任何内容（包括注释）。

11.3.3　元素

元素是 XML 文档的主体，也是 XML 文档的基本单位。一个元素以起始标记开始、以结束标记终止，而元素的内容则置于起始标记与结束标记之间。如以下即为一个元素的示例（book 元素）：

```
<book>SCO UNIX(OpenServer)系统管理与解决方案</book>
```

每个 XML 文档都必须有而且只能有一个根元素。在根元素中，可以根据需要包含有其他一个或多个子元素。此外，在某个子元素中，还可以包含有其他一个或多个子元素。这样，便构成了元素之间的嵌套关系。所谓元素的嵌套，就是指一个元素（除根元素外）必须完全包含在另一个元素之中，而不能出现互相重叠的情况。

【例 11.4】 元素嵌套示例（11-4.XML）。

```
<?xml version="1.0" encoding="GB2312" standalone="yes"?>
<book>
 <title>PowerBuilder 数据库应用开发技术</title>
 <author>卢守东</author>
 <publisher>清华大学出版社</publisher>
 <date>
  <year>2006</year>
  <month>6</month>
  <day>1</day>
 </date>
 <isbn>7-302-12729-8</isbn>
</book>
```

在该文档中，子元素 date 又包含有三个子元素，即分别表示年、月、日的 year、month、day。在 IE 浏览器中打开该文档（其文件名为 11-4.XML），显示结果如图 11.4 所示。对于包含有子元素的元素，在 IE 浏览器中显示时是可折叠的。

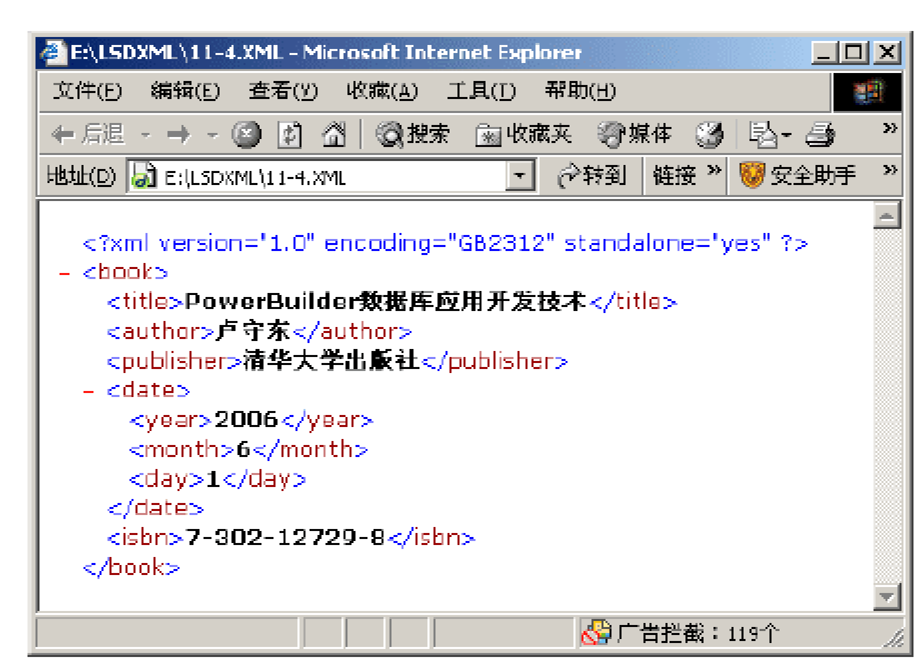


图 11.4 在 IE 中打开 11-4.XML 的结果

通常，可用父子关系来描述 XML 文档中元素的嵌套关系，而具有相同父元素的子元素则互称为兄弟元素。在一个 XML 文档中，每一个子元素都必须完全包含在其父元素中，兄弟元素之间是不能相互重叠的。为形象直观起见，一般可用相应的树状结构来表示出 XML 文档的逻辑结构。如图 11.5 所示，即为例 11.4 中 XML 文档的树状结构（或称之为结构树）。

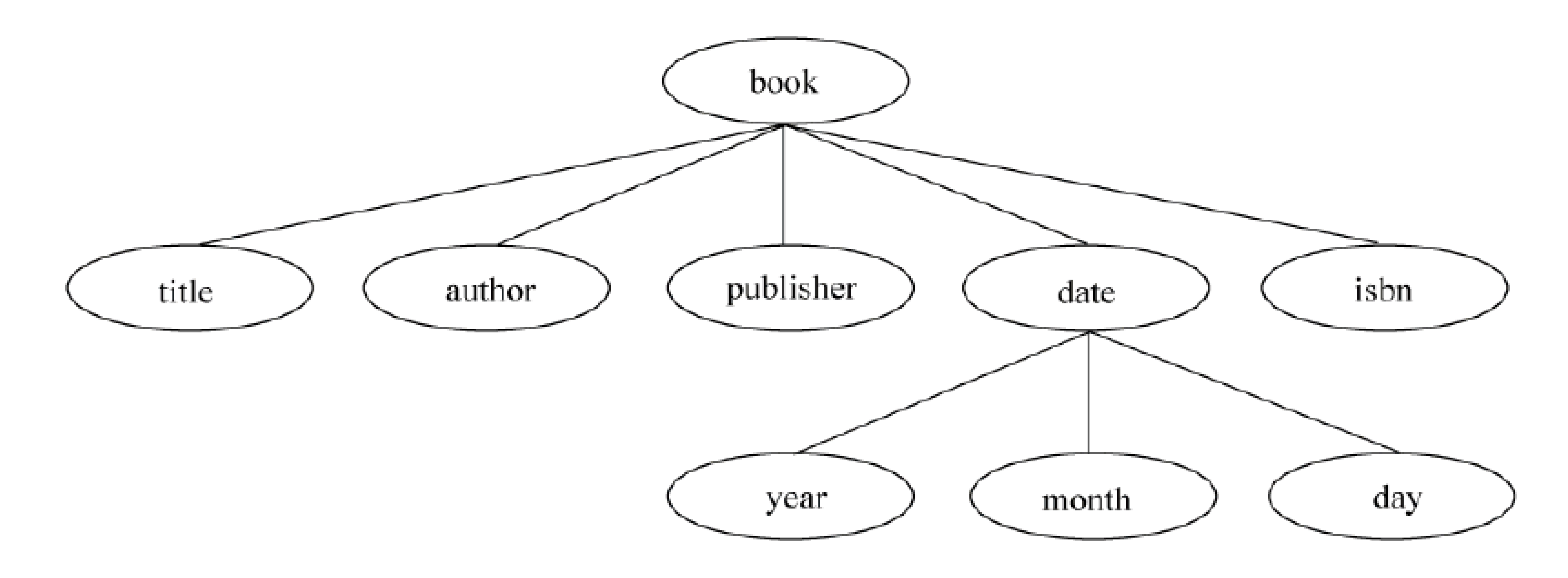


图 11.5 文档 11-4.XML 的树状结构

【例 11.5】 元素重叠示例（11-5.XML）。

```
<?xml version="1.0" encoding="GB2312" standalone="yes"?>
<book>
 <title>PowerBuilder 数据库应用开发技术</title>
 <author>卢守东<publisher>清华大学出版社</author></publisher>
```

```
</book>
```

在该文档中，元素 author 与 publisher 相互重叠，因此是不允许的。

在 XML 文档中，还允许使用空元素，即不包含有任何内容的元素。实际上，一个空元素就是将一个起始标记紧接着其结束标记。如以下即为一个空元素的示例（book 空元素）：

```
<book></book>
```

空元素还有另外一种等价的写法，即不用结束标记，而将起始标记中的“>”改为“/>”（在“/”前最好加上一个空格）。如以下示例：

```
<book/>
```

11.3.4 属性

属性是依附于元素而存在的。与 HTML 一样，在 XML 的起始标记中也可以包含有一个或多个属性，且各个属性均以空格分隔开，而每个属性则是以等号（=）分隔的属性名与属性值对。

【例 11.6】 元素属性示例（11-6.XML）。

```
<?xml version="1.0" encoding="GB2312" standalone="yes"?>
<book author="卢守东" publisher="清华大学出版社" date="2006-6-1">
PowerBuilder 数据库应用开发技术
</book>
```

在该示例中，book 元素共有三个属性，即 author 属性，其值为“卢守东”；publisher 属性，其值为“清华大学出版社”；date 属性，其值为“2006-6-1”。

在 XML 中，属性名遵循与标记名一样的命名规则，即属性名必须以字母或下划线（_）开头，后面的字符可以是字母、数字、下划线（_）、短横线（-）或小圆点（.），且不能包含有空格。应该注意的是，在同一个标记中，不能出现同名的属性。此外，与标记名一样，属性名也是区分大小写的。如 author 属性与 Author 属性是两个不同的属性。

与 HTML 不同，在 XML 中，属性值必须以单引号（' '）或双引号（" "）来界定，即使用单引号或双引号括起来。通常，若属性值本身包含有单引号，则以双引号来界定；反之，若属性值本身包含有双引号，则以单引号来界定。特别地，如果属性值本身包含有单引号与双引号，那么属性值中的引号就必须用相应的实体（稍后将介绍）来代替。如以下示例：

```
<message content='He said, "That&apos;s OK!"' />
```

在该示例中，content 属性的值为“He said, "That's OK!"”。其中，“That's OK!”中的“'”在此以实体引用“'”来代替。

11.3.5 注释

与 HTML 一样，在 XML 中也可以添加注释，且具有相同的格式。所谓注释，实际上

是一些说明性的文字，目的是增强文档的可读性，使其更加清晰易懂。在处理时，XML 解析器会忽略所有的注释内容。

注释以“<!--”开始、以“-->”结束，其语法格式为：

```
<!-- 注释的内容 -->
```

注释可以独占一行，也可以分为多行书写。对于分为多行书写的注释，在 IE 浏览器中显示时是可折叠的。在注释的内容中，不能再使用“--”符号。此外，也不能在注释中再嵌套注释。

【例 11.7】 注释使用示例（11-7.XML）。

```
<?xml version="1.0" encoding="GB2312" standalone="yes"?>
<!--根元素开始标记-->
<book>
 <!--子元素 title-->
 <title>PowerBuilder 数据库应用开发技术</title>
 <!--子元素 author-->
 <author>卢守东</author>
 <!--子元素 publisher-->
 <publisher>清华大学出版社</publisher>
</book> <!--根元素结束标记-->
```

注释不是 XML 的元素，因此可置于根元素之前或之后，但不能位于 XML 声明之前。此外，注释也不能放在标记之中。否则，会破坏标记的完整性。

11.3.6 实体

一个 XML 文档可分为标记与内容两大部分，并以相应的标记字符作为识别依据。在 XML 中，标记字符共有五个，分别为“<”（小于号）、“>”（大于号）、“&”（与符号）、“'”（单引号）与“"”（双引号）。在处理时，即使这些特殊字符（特别是“<”与“&”）出现在内容之中，XML 解析器也会将其作为标记字符看待，而非普通内容。如果在内容中要将这些标记字符作为普通字符处理，那么就必须引用相应的实体。

所谓实体，实际上就是一些在分析或处理文档时会被相应字符或数据所取代的特定的标识。在 XML 中，已针对标记字符预定义了五个相应的字符实体，其引用方式及对应的字符如表 11.1 所示。

表 11.1 XML 的预定义字符实体

实体引用	字符	实体引用	字符
&	&	'	'
<	<	"	"
>	>		

在 XML 中引用实体时，应注意以“&”开始，并以“;”结束。实体的引用既可以出现在元素的内容中，也可以出现在标记的属性值中。

【例 11.8】 实体引用示例（11-8.XML）。

```
<?xml version="1.0" encoding="GB2312" standalone="yes"?>
<example>
 <title>XML 文档实例</title>
 <content>
 &lt;?xml version="1.0" encoding="GB2312" standalone="yes"?&gt;
 &lt;book&gt;
 &lt;title&gt;PowerBuilder 数据库应用开发技术&lt;/title&gt;
 &lt;author&gt;卢守东&lt;/author&gt;
 &lt;publisher&gt;清华大学出版社&lt;/publisher&gt;
 &lt;/book&gt;
 </content>
</example>
```

在该文档中，子元素 content 的内容实际上是一个 XML 文档，其中包含有相应的字符实体引用。在 IE 浏览器中打开该文档（其文件名为 11-8.XML），显示结果如图 11.6 所示。

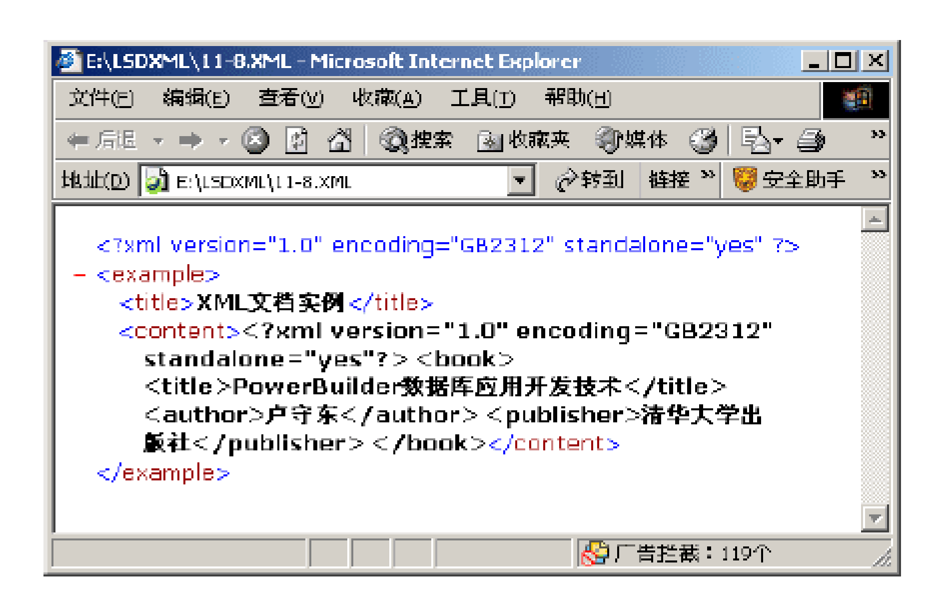


图 11.6　在 IE 中打开 11-8.XML 的结果

11.3.7　CDATA 段

在 XML 文档的元素内容中，如果包含有要作为普通字符处理的标记字符，那么就必须以相应的实体引用来代替。但如果所包含的标记字符数量众多（如程序、表达式等），那么要逐一地进行替换，显然就不是一件轻易的事了。在这种情况下，可考虑在 XML 文档中插入相应的 CDATA 段。

CDATA 段的语法格式为：

```
<![CDATA[
文本内容
]]>
```

CDATA 段以“<![CDATA[”开始、以“]]>”结束，二者之间则为相应的文本内容。其中，“CDATA”必须为大写形式。

CDATA 段是 XML 文档中的特殊区域，其中所包含的整个文本内容均会解释为纯字符数据，即使是标记字符也不例外。

【例 11.9】 CDATA 段使用示例（11-9.XML）。

```
<?xml version="1.0" encoding="GB2312" standalone="yes"?>
<example>
 <title>XML 文档实例</title>
 <content>
<![CDATA[
<?xml version="1.0" encoding="GB2312" standalone="yes"?>
<book>
```

```
  <title>PowerBuilder 数据库应用开发技术</title>
  <author>卢守东</author>
  <publisher>清华大学出版社</publisher>
</book>
]]>
</content>
</example>
```

在 IE 浏览器中打开该文档（其文件名为 11-9.XML），显示结果如图 11.7 所示。

```
<?xml version="1.0" encoding="GB2312" standalone="yes" ?>
- <example>
    <title>XML文档实例</title>
  - <content>
    - <![CDATA[
        <?xml version="1.0" encoding="GB2312" standalone="yes"?>
        <book>
         <title>PowerBuilder数据库应用开发技术</title>
         <author>卢守东</author>
         <publisher>清华大学出版社</publisher>
        </book>
      ]]>
    </content>
  </example>
```

图 11.7　在 IE 中打开 11-9.XML 的结果

应该注意的是，在 CDATA 段所包含的文本内容中，是不允许出现“]]>”的。正因为如此，CDATA 段是不能嵌套的。

11.4　XML 文档的类型定义

遵循 XML 语法规则的 XML 文档通常称为格式良好的（well-formed） XML 文档（或良构的 XML 文档）。但格式良好的 XML 文档不一定就是有效的（valid）XML 文档（或合法的 XML 文档）。一个有效的 XML 文档，除了要遵循 XML 的语法规则以外，还必须符合相应的文档结构定义。文档的结构定义其实就是文档的建模，而 XML 最先使用的一种文档建模方法即为文档类型定义（document type definition，DTD）。在此，将对 DTD 建模方式进行相应的介绍。

11.4.1　DTD 简介

在 XML 中，用户可自行创建所需要的标记以及标记中的属性。为指定 XML 文档的逻辑结构，并检验文档中标记与属性的使用是否正确，XML 便引入了文档类型定义 DTD。

一个 DTD 实际上就是一类 XML 文档的结构定义或构成规则，声明了在该类文档中所能包含的元素、属性、实体及其相互之间的关系。通过 DTD，可严格规定以其作为蓝本所

创建的所有 XML 实例文档的树状层次结构的全部细节。例如，使用一个 DTD 可以指定其文档实例的根元素为 book 元素，且 book 元素包含有 title、author、publisher、date、isbn 五个子元素。

DTD 有内部 DTD 与外部 DTD 之分。其中，内部 DTD 直接包含在相应的 XML 文档中，而外部 DTD 则保存在一个独立的 DTD 文件中。相比之下，由于外部 DTD 是以文件的形式而存在的，因此可被不同的 XML 文档引用或共享。由此可见，使用外部 DTD，可为不同的应用定义一个共同的数据组织标准。

一旦一个 XML 文档引用了某个 DTD，有关的分析程序便可对二者进行相应的比较。XML 文档与相应 DTD 比较的过程其实就是根据 DTD 中所指定的约束条件来检验 XML 文档中标记等的使用是否合法的过程，通常称之为 XML 文档的合法性检验（或有效性检验）。若文档符合 DTD 的约束，则该文档就被认为是合法的（或有效的），否则就被认为是非法的（或无效的）。

应该注意的是，多数 Web 浏览器（包括 IE 6.0 等）并不检查 XML 文档的合法性。但在 Internet（因特网）上可以找到很多能进行合法性检验的分析程序，且大多数都是免费的，如 IBM 的 AlphaWorks XML for Java、Microsoft 与 DataChannel 的 XJParser、Silfide 的 SXP 等。

11.4.2 DTD 的声明

在 XML 文档中，对于 DTD 的引用是通过相应的 DTD 声明来实现的。DTD 声明一般位于 XML 声明与文档主体之间。实际上，XML 声明与 DTD 声明均属于 XML 文档的序言部分。

对于内部 DTD 与外部 DTD，其声明的格式略有不同。下面分别进行介绍。

1. 内部 DTD 的声明

在 XML 文档中，声明内部 DTD 的语法格式为：

```
<!DOCTYPE 根元素名称 [
    DTD 的具体定义
]>
```

内部 DTD 声明以“<!DOCTYPE”开始、以“]>”结束，二者之间则为 DTD 的具体定义。其中，“DOCTYPE”必须为大写形式，而 DTD 的具体定义则包括 DTD 的元素声明、属性声明与实体声明等。在 DTD 的声明中，必须指定相应 XML 文档根元素的名称。

【例 11.10】 包含内部 DTD 的 XML 文档示例（11-10.XML）。

```
<?xml version="1.0" encoding="gb2312" standalone="yes"?>
<!DOCTYPE book[
 <!ELEMENT book (name,author,publisher,date,isbn)>
 <!ELEMENT name (#PCDATA)>
 <!ELEMENT author (#PCDATA)>
 <!ELEMENT publisher (#PCDATA)>
 <!ELEMENT date (#PCDATA)>
 <!ELEMENT isbn (#PCDATA)>
]>
```

```
<book>
 <name>PowerBuilder 数据库应用开发技术</name>
 <author>卢守东</author>
 <publisher>清华大学出版社</publisher>
 <date>2006-06-01</date>
 <isbn>7-302-12729-8</isbn>
</book>
```

在该示例的 DTD 中，声明了文档的根元素为 book，并指定其子元素依次为 name、author、publisher、date 与 isbn，且各子元素的内容均为可解析的字符数据（即纯文本内容）。在 IE 浏览器中打开该文档（其文件名为 11-10.XML），显示结果如图 11.8 所示。

在 IE 浏览器中，DTD 的声明被显示为“<!DOCTYPE 根元素名称(View Source for full doctype...)>”。

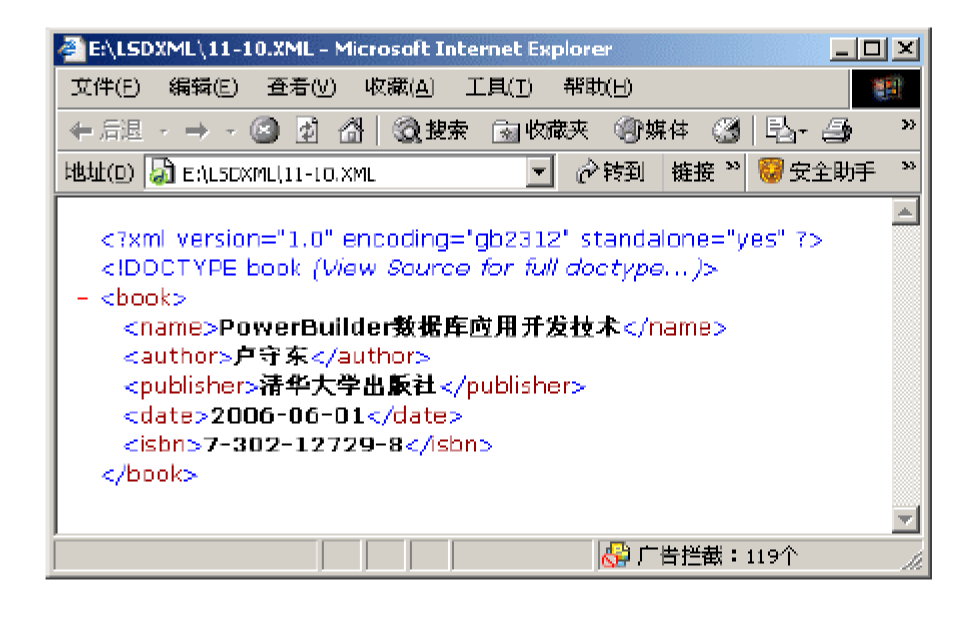


图 11.8　在 IE 中打开 11-10.XML 的结果

2．外部 DTD 的声明

外部 DTD 与内部 DTD 不同，通常是以扩展名为“.DTD”的单独文件的形式而存在的。根据其使用范围或是否经过标准化，外部 DTD 又可分为私有 DTD 与公共 DTD 两种，二者的声明格式也是有所不同的。

1）私有 DTD 的声明

私有 DTD 是指由个人或工作小组使用的、未经权威机构标准化的 DTD，其使用范围要相对小一些。

在 XML 文档中，私有 DTD 是使用 SYSTEM 关键字（大写形式）来进行声明的，其语法格式为：

```
<!DOCTYPE 根元素名称 SYSTEM "systemId">
```

其中，systemId 为是所要引用的私有 DTD 文件系统标识符（system identifier），其实就是相应的 URL 地址，可以是绝对路径的形式，也可以是相对路径的形式。特别地，如果 DTD 文件与相应的 XML 文档处于同一个目录下，那么 systemId 就可以直接指定为该 DTD 文件的文件名。

【例 11.11】 引用外部私有 DTD 的 XML 文档示例（11-11.XML）。

```
<?xml version="1.0" encoding="gb2312" standalone="no"?>
<!DOCTYPE book SYSTEM "11-11.DTD">
<book>
 <name>PowerBuilder 数据库应用开发技术</name>
 <author>卢守东</author>
 <publisher>清华大学出版社</publisher>
 <date>2006-06-01</date>
 <isbn>7-302-12729-8</isbn>
</book>
```

其中，私有 DTD 文件 11-11.DTD 的内容为：

```
<?xml version="1.0" encoding="gb2312" ?>
<!ELEMENT book (name,author,publisher,date,isbn)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT isbn (#PCDATA)>
```

应该注意的是，为引用外部 DTD，应在 XML 声明中说明当前文档不是独立的（或自成一体的）。为此，只需将 XML 声明中的 standalone 属性的值设置为 no 即可。

2）公共 DTD 的声明

公共 DTD 是指由公众或特定行业使用的、已经权威机构标准化的 DTD，其使用范围要更为广泛一些。

在 XML 文档中，公共 DTD 是使用 PUBLIC 关键字（大写形式）来进行声明的。实际上，公共 DTD 的声明方式与私有 DTD 的声明方式相类似，除了使用 PUBLIC 代替 SYSTEM 以外，只需再指定相应的公共标识符（public identifier）即可。

声明公共 DTD 的语法格式为：

```
<!DOCTYPE 根元素名称 PUBLIC "publicId" "systemId">
```

其中，publicId 即为所要引用的公共标识符，其实就是相应的公共 DTD 的名称，用于在中心数据库（通常是标准化组织的 DTD 数据库）识别相应的公共 DTD 文件。若分析程序无法根据 publicId 所指定的名称从中心数据库搜索到相应的 DTD 文件，则会再到 systemId（系统标识符）所指定的位置查找相应的 DTD 文件。

作为一个正式的公共标识符，公共 DTD 的名称只能包含有字母、数字、空格以及其他一些符号（-、%、$、#、@、(、)、+、:、=、/、!、*、;、?），其一般格式为：

```
开始字符//所有者的名称//所描述的文件类型//所使用的语言种类
```

其中，开始字符为“ISO”、“+”（加号）、“–”（减号）三者之一，所有者的名称与所描述的文件类型均为相应的简短的字符串，所使用的语言种类则为 ISO639 标准所规定的语言标识符之一（如中文为 ZH、英文为 EN 等）。对于 ISO 标准的 DTD，其名称应以“ISO”作为开始字符；对于非 ISO 标准组织的 DTD，其名称应以“+”作为开始字符；对于非标准组织的 DTD，其名称应以“–”作为开始字符。如以下声明示例：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
       "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

在该声明中，说明了 XHTML 文档（1.1 版）的根元素是 html，并在名称为“-//W3C//DTD XHTML 1.1//EN”的公共 DTD 中进行了定义。应用程序（如浏览器等）可据此名称查找相应的公共 DTD。若找不到，则使用此名称后面的 URL 作为查找位置。

3．内部 DTD 与外部 DTD 的混合声明

在 XML 文档中，可以同时进行内部 DTD 与外部 DTD 的声明。

内部 DTD 与私有 DTD 的混合声明格式为：

```
<!DOCTYPE 根元素名称 SYSTEM "systemId" [
    内部 DTD 的具体定义
]>
```

内部 DTD 与公共 DTD 的混合声明格式为：

```
<!DOCTYPE 根元素名称 PUBLIC "publicId " "systemId" [
    内部 DTD 的具体定义
]>
```

【例 11.12】 内部 DTD 与外部 DTD 的混合声明示例（11-12.XML）。

```
<?xml version="1.0" encoding="gb2312" standalone="no"?>
<!DOCTYPE book SYSTEM "11-12.DTD" [
 <!ELEMENT name (#PCDATA)>
 <!ELEMENT author (#PCDATA)>
 <!ELEMENT publisher (#PCDATA)>
 <!ELEMENT date (#PCDATA)>
 <!ELEMENT isbn (#PCDATA)>
]>
<book>
 <name>PowerBuilder 数据库应用开发技术</name>
 <author>卢守东</author>
 <publisher>清华大学出版社</publisher>
 <date>2006-06-01</date>
 <isbn>7-302-12729-8</isbn>
</book>
```

其中，私有 DTD 文件 11-12.DTD 的内容为：

```
<?xml version="1.0" encoding="gb2312" ?>
<!ELEMENT book (name,author,publisher,date,isbn)>
```

实际上，内部 DTD 与外部 DTD 是互为补充的。若二者出现冲突，则内部 DTD 优先。在进行合法性检验时，若在内部 DTD 中找不到声明，则会自动到外部 DTD 中去查找。

通常，可将公用的声明置于外部 DTD 中，而将专用的声明置于内部 DTD 中。这样，在保护内部 DTD 的同时，又有利于外部 DTD 的共享及其升级或更新。

11.4.3 DTD 的元素声明

在 DTD 中，元素是使用 ELEMENT 关键字（大写形式）来进行声明的，其基本格式为：

```
<!ELEMENT element_name content_model>
```

其中，element_name 为元素的名称，实际上是一个合法的标记名；content_model 为元素的内容模型，可根据实际情况指定为不同的类型——EMPTY 型、ANY 型、#PCDATA 型、子元素型或混合型。

1．ANY 型元素的声明

ANY 型元素的内容类型为“ANY”（大写形式），其声明格式为：

```
<!ELEMENT element_name ANY>
```

作为 ANY 型元素，其内容是不受任何限制的，可以为空，也可以是字符数据，还可以是任何类型的子元素。ANY 型元素通常用于未经结构化的文档中。

在 DTD 中，首先要声明根元素。若暂时无法确定其内容的类型，则可将其声明为 ANY 型。

【例 11.13】 ANY 型元素的声明示例（11-13.XML）。

```
<?xml version="1.0" encoding="gb2312" standalone="yes"?>
<!DOCTYPE book[
 <!ELEMENT book ANY>
]>
<book>PowerBuilder 数据库应用开发技术</book>
```

该文档是合法的。其中，根元素 book 的内容为字符数据。

【例 11.14】 ANY 型元素的声明示例（11-14.XML）。

```
<?xml version="1.0" encoding="gb2312" standalone="yes"?>
<!DOCTYPE book[
 <!ELEMENT book ANY>
]>
<book>
 <name>PowerBuilder 数据库应用开发技术</name>
 <author>卢守东</author>
</book>
```

在该文档中，根元素 book 被声明为 ANY 型，即可包含其他任意元素。但另一方面，根元素 book 所包含的 name 子元素与 author 子元素却并未在 DTD 中加以声明，故不符合 DTD 的约束条件。换言之，该文档是非法的。

由于 ANY 型元素破坏了 DTD 的严格性，因此建议不要随意使用。

2．EMPTY 型元素的声明

EMPTY 型元素的内容类型为“EMPTY”（大写形式），其声明格式为：

```
<!ELEMENT element_name EMPTY>
```

作为 EMPTY 型元素，其内容为空，即不能包含有字符数据或子元素。对于 EMPTY 型元素来说，相关的数据只能通过相应的属性来表示。

在 XML 文档中，如果包含有空元素，那么在相应的 DTD 中就应该将其声明为 EMPTY 型元素。如以下示例：

```
<!ELEMENT nothing EMPTY>
```

在该示例中，元素 nothing 被声明为一个 EMPTY 型元素。

3．#PCDATA 型元素的声明

#PCDATA 型元素的内容类型为“#PCDATA”（大写形式），其声明格式为：

```
<!ELEMENT element_name (#PCDATA)>
```

作为#PCDATA 型元素，其内容为可解析的字符数据（parsed character data），即非标记文本。#PCDATA 型元素是不能包含有任何其他子元素的。

【例 11.15】 #PCDATA 型元素的声明示例（11-15.XML）。

```
<?xml version="1.0" encoding="gb2312" standalone="yes"?>
<!DOCTYPE book[
 <!ELEMENT book (#PCDATA)>
]>
<book>PowerBuilder 数据库应用开发技术</book>
```

4. 子元素型元素的声明

子元素型元素的内容为相应的子元素列表，其声明格式为：

```
<!ELEMENT element_name(subelement_list)>
```

其中，圆括号“()”内的 subelement_list 即为子元素列表，由一个或多个子元素构成。子元素列表中的各个子元素之间一般以逗号（,）作为分隔符，其顺序代表了相应 XML 文档中各个子元素的出现顺序。如果要在多个子元素之间选择其中之一，那么应以竖线（|）作为分隔符。

【例 11.16】 子元素型元素的声明示例（11-16.XML）。

```
<?xml version="1.0" encoding="gb2312" standalone="yes"?>
<!DOCTYPE book[
 <!ELEMENT book (name,author)>
 <!ELEMENT name (#PCDATA)>
 <!ELEMENT author (#PCDATA)>
]>
<book>
 <name>PowerBuilder 数据库应用开发技术</name>
 <author>卢守东</author>
</book>
```

在该示例中，子元素 name 与 author 必须依次出现。

【例 11.17】 子元素型元素的声明示例（11-17.XML）。

```
<?xml version="1.0" encoding="gb2312" standalone="yes"?>
<!DOCTYPE book[
 <!ELEMENT book (name,(author|publisher))>
 <!ELEMENT name (#PCDATA)>
 <!ELEMENT author (#PCDATA)>
 <!ELEMENT publisher (#PCDATA)>
]>
<book>
 <name>PowerBuilder 数据库应用开发技术</name>
 <publisher>清华大学出版社</publisher>
</book>
```

在该示例中，子元素 author 与 publisher 只能出现其中之一，而不能同时出现。

在声明子元素型的元素时，还可以同时指定相应子元素的出现次数。为此，只需在子元素的名称后面加上相应的控制字符即可。可供使用的子元素控制字符及其含义如表 11.2 所示。

表 11.2　子元素控制字符及其含义

控制字符	含义
?	出现 0 次或 1 次，即最多出现一次
+	出现 1 次或多次，即至少出现一次
*	出现 0 次、1 次或多次，即可出现任意次

【例 11.18】 子元素控制字符的使用示例（11-18.XML）。

```
<?xml version="1.0" encoding="gb2312" standalone="yes"?>
<!DOCTYPE book[
 <!ELEMENT book (name,subname?,author*,publisher+,date)>
 <!ELEMENT name (#PCDATA)>
 <!ELEMENT subname (#PCDATA)>
 <!ELEMENT author (#PCDATA)>
 <!ELEMENT publisher (#PCDATA)>
 <!ELEMENT date (#PCDATA)>
]>
<book>
 <name>Visual FoxPro 数据库教程与实验</name>
 <author>徐辉</author>
 <author>卢守东</author>
 <author>聂良刚</author>
 <author>庞大连</author>
 <publisher>清华大学出版社</publisher>
 <date>2005-01-01</date>
</book>
```

在该示例的 DTD 中，子元素 subname（表示副书名）被声明为最多出现一次（即不出现或只出现一次），子元素 author 被声明为可出现任意次，子元素 publisher 被声明为至少出现一次。

应该注意的是，使用控制字符“+”与“*”的子元素在相应的 XML 文档中均可出现多次，但其多次出现必须是连续的，否则是非法的。

在声明子元素型元素时，还可以采用枚举的办法在子元素列表中列出需要多次出现的子元素。

【例 11.19】 子元素的枚举声明示例（11-19.XML）。

```
<?xml version="1.0" encoding="gb2312" standalone="yes"?>
<!DOCTYPE 编著情况[
```

```
 <!ELEMENT 编著情况 (姓名,代表作,代表作)>
 <!ELEMENT 姓名 (#PCDATA)>
 <!ELEMENT 代表作 (#PCDATA)>
]>
<编著情况>
 <姓名>卢守东</姓名>
 <代表作>SCO UNIX(OpenServer)系统管理与解决方案</代表作>
 <代表作>PowerBuilder 数据库应用开发技术</代表作>
</编著情况>
```

在该示例中，子元素“代表作”必须出现两次。

5. 混合型元素的声明

混合型元素的内容可以是字符数据与子元素的任意组合，其声明格式为：

```
<!ELEMENT element_name(#PCDATA|subelement_list)*>
```

其中，subelement_list 为以竖线(|)作为分隔符的子元素列表。在圆括号“()”中，“#PCDATA”必须置于最前面。在圆括号“()”后，必须加上不限次数的控制字符“*”。

【例 11.20】 混合型元素的声明示例（11-20.XML）。

```
<?xml version="1.0" encoding="gb2312" standalone="yes"?>
<!DOCTYPE p[
 <!ELEMENT p (#PCDATA|b|i)*>
 <!ELEMENT b (#PCDATA)>
 <!ELEMENT i (#PCDATA)>
]>
<p>
 This <i>is</i> an <b>example</b> of <i>mixed</i> <i>content</i>!
</p>
```

11.4.4 DTD 的属性声明

在 DTD 中，元素的属性是使用 ATTLIST 关键字（大写形式）来进行声明的，其基本格式为：

```
<!ATTLIST element_name
    attribute_name_1 attribute_type default_set
    attribute_name_2 attribute_type default_set
    ⋮
    attribute_name_n attribute_type default_set>
```

其中，element_name 为元素的名称，attribute_name_1 ~ attribute_name_n 为元素各属性的名称，attribute_type 与 default_set 则为相应属性的类型与默认设置。

属性的类型用于约束属性的取值，各属性类型及其说明如表 11.3 所示。

表 11.3　属性的类型及其说明

类型	说明
CDATA	属性值为不含标记的任意字符数据，即纯文本内容。CDATA 类型是最常用的一种属性类型
ID	属性值是一个合法的 XML 名称（即应符合 XML 中名称的命名规则），且必须是唯一的。一个元素只能有一个 ID 类型的属性，ID 类型的属性可用于唯一地标识文档中的有关元素
IDREF	属性值是文档中其他某个元素的 ID 类型属性值，可用于建立当前元素与指定元素之间的关系
IDREFS	属性值是用空格分开的其他有关元素的 ID 类型属性值，可用于建立当前元素与其他元素之间的一对多关系
ENTITY	属性值是一个在 DTD 中声明的非解析实体的名称，可用于引用相应的外部实体
ENTITIES	属性值是用空格分开的多个在 DTD 中声明的非解析实体的名称，可用于引用相应的外部实体
NMTOKEN	属性值是一个合法的 XML 名称
NMTOKENS	属性值是用空格分开的多个合法的 XML 名称
NOTATION	属性值是一个在 DTD 中声明的记号
Enumeration	枚举类型，即置于圆括号“()”之中的以竖线“|”分开的所有可能的属性取值

属性的默认设置用于指定属性的默认值或指定该属性是否为必需的属性。各种可能的属性默认设置及其说明如表 11.4 所示。

表 11.4　属性的默认设置及其说明

类型	说明
"aValue"	该属性的默认值（在此以“aValue”表示）。在 XML 文档的相应元素中，若未对该属性赋值，则取其在 DTD 中所指定的默认值
#REQUIRED	该属性对于指定的元素来说是必需的
#IMPLIED	该属性对于指定的元素来说是可选的，即可以出现也可以不出现
#FIXED "aValue"	该属性的值必须为指定的默认值（在此以“aValue”表示）。在 XML 文档的相应的元素中，若未使用该属性，则自动包含并取其默认值；若使用该属性，则其属性值必须指定为相应的默认值

【例 11.21】 属性声明示例（11-21.XML）。

```
<?xml version="1.0" encoding="gb2312" standalone="yes"?>
<!DOCTYPE family[
 <!ELEMENT family (member*)>
 <!ELEMENT member (#PCDATA)>
 <!ATTLIST member
              memberid ID #REQUIRED
              father IDREF #IMPLIED
              mother IDREF #IMPLIED>
]>
<family>
 <member memberid="M01">卢强</member>
 <member memberid="M02">刘芳</member>
```

```
 <member memberid="M03" father="M01" mother="M02">卢铭</member>
</family>
```


11.4.5 DTD 的实体声明

在 DTD 中，实体（entity）是使用“ENTITY”关键字（大写形式）来进行声明的。实体被声明后，即可通过实体引用应用于 DTD 或 XML 文档中。在此所说的实体，是指用户自定义的实体，而并非 XML 预定义的字符实体。预定义的字符实体共有五个，无须声明即可直接引用。

XML 中的实体类似于一般程序设计语言中的符号常量。每个实体都有相应的名称，并代表了某些相关的内容。在 DTD 或 XML 文档中所引用的实体，在分析时将被其内容所替代。实体只需声明一次，即可多次重复引用。这对于 DTD 或 XML 文档的设计与维护来说，都是极为有利的。

对于一个给定的实体来说，不是通用的（general）就是参数的（parameter），不是内部的（internal）就是外部的（external），不是可解析的（parsed）就是非解析的（unparsed）。其中，通用实体只能在 XML 文档中被引用（而不能在 DTD 中被引用），参数实体只能在 DTD 中被引用（而不能在 XML 文档中被引用）；内部实体的值包含在 DTD 中，外部实体的值包含在外部资源中；可解析实体的值为可以解析的数据（如字符数据等），非解析实体的值为不可解析的数据（如二进制文件、图像文件等）。其实，非解析实体总是通用的和外部的，而参数实体、内部实体则总是可解析的。

实际上，一共只有五种不同类型的实体（如图 11.9 所示），即内部通用实体、外部可解析通用实体、外部非解析通用实体、内部参数实体与外部参数实体。

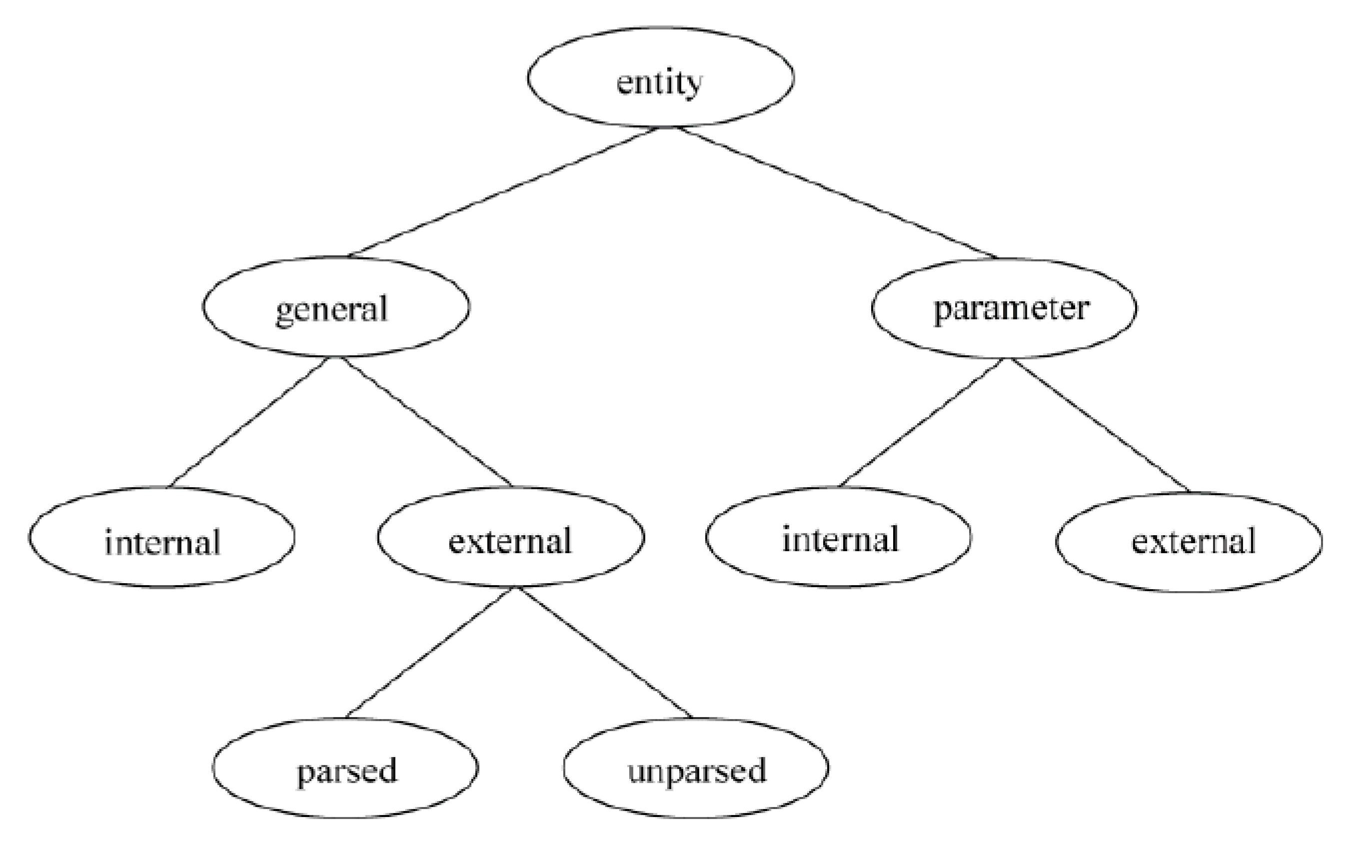


图 11.9　实体的分类

1．内部通用实体

内部通用实体的声明格式为：

```
<!ENTITY name "value">
```

其中，name 为实体的名称，value 为实体所代表的可解析的内容。

作为内部通用实体，只能在 DTD 中进行声明，并且只能在 XML 文档中被引用。内部通用实体总是包含了可解析的内容。在 XML 文档中，一个到内部通用实体的引用（&name;）将被替换为解析后的内容。

【例 11.22】 内部通用实体的使用示例（11-22.XML）。

```
<?xml version="1.0" encoding="gb2312" standalone="yes"?>
<!DOCTYPE book[
 <!ELEMENT book (name,author,other)>
 <!ELEMENT name (#PCDATA)>
 <!ELEMENT author (#PCDATA)>
 <!ELEMENT other ANY>
 <!ENTITY lsd "卢守东">
 <!ENTITY other "<other>LSD690@163.com</other>">
]>
<book>
 <name>PowerBuilder 数据库应用开发技术</name>
 <author>&lsd;</author>
 &other;
</book>
```

在 IE 浏览器中打开该文档（其文件名为 11-22.XML），显示结果如图 11.10 所示。

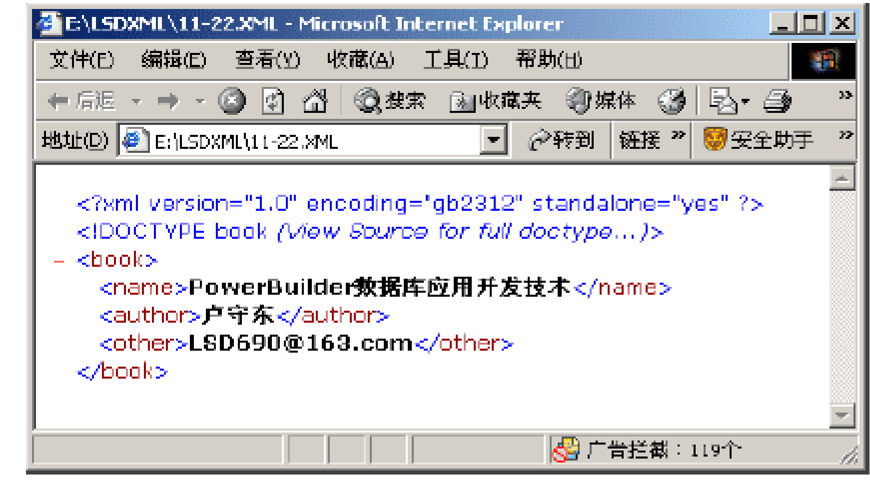


图 11.10　在 IE 中打开 11-22.XML 的结果

2．外部可解析通用实体

外部可解析通用实体的声明格式为：

```
<!ENTITY name SYSTEM "systemId">
<!ENTITY name PUBLIC "publicId" "systemId">
```

其中，name 为实体的名称，systemId 与 publicId 为包含实体内容的外部资源的系统标识符（system identifier）与公共标识符（public identifier）。

除了实体所代表的内容包含在某个外部资源之中以外，外部可解析通用实体的使用方法与内部通用实体的使用方法是完全相同的，也只能在 XML 文档中被引用。外部可解析通用实体总是包含了可解析的内容。在 XML 文档中，一个到外部可解析通用实体的引用（&name;）将被替换为解析后的内容。

【例 11.23】 外部可解析通用实体的使用示例（11-23.XML）。

```
<?xml version="1.0" encoding="gb2312" standalone="no"?>
<!DOCTYPE book[
 <!ELEMENT book (name,author,content)>
 <!ELEMENT name (#PCDATA)>
 <!ELEMENT author (#PCDATA)>
 <!ELEMENT content (#PCDATA)>
 <!ENTITY content SYSTEM "11-23.txt">
]>
<book>
```

```
  <name>PowerBuilder 数据库应用开发技术</name>
  <author>卢守东</author>
  <content>&content;</content>
</book>
```

其中，外部资源文件 11-23.txt 的内容为：

```
<?xml version="1.0" encoding="gb2312" ?>
```

本书以 PowerBuilder 9.0 为蓝本，结合多年的教学实践与应用系统开发的实际经验，全面介绍 PowerBuilder 数据库应用系统的开发步骤、方法与技术。

在 IE 浏览器中打开该文档（其文件名为 11-23.XML），显示结果如图 11.11 所示。

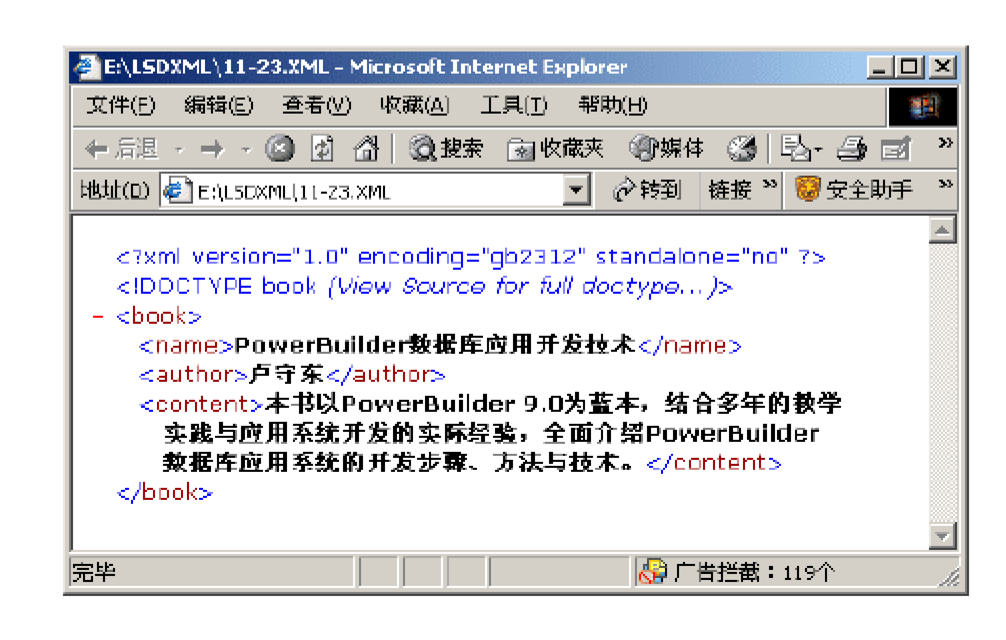


图 11.11　在 IE 中打开 11-23.XML 的结果

3．外部非解析通用实体

外部非解析通用实体的声明格式为：

```
<!ENTITY name SYSTEM "systemId"
NDATA nname>
<!ENTITY name PUBLIC "publicId" "systemId" NDATA nname >
```

其中，name 为实体的名称，systemId 与 publicId 为包含实体内容的外部资源的系统标识符与公共标识符，nname 为实体的符号名称（notation name）。符号名称通常用于指定相应资源的类型或格式。

外部非解析通用实体通常又简称为非解析实体。其实，非解析实体总是通用的和外部的。通过使用非解析实体，可向 XML 文档附加任意的二进制资源。实际上，非解析实体只是简单地通过相应的公共标识符或系统标识符指向资源。至于如何处理非解析实体，则依赖于相应的处理程序。由于非解析实体可以引用任何二进制资源，因此处理程序需要符号名称所提供的附加信息来决定资源的类型。

与其他通用实体的引用不同，非解析实体的引用是通过 ENTITY 或 ENTITIES 类型的属性来实现的。

【例 11.24】 外部非解析通用实体的使用示例（11-24.XML）。

```
<?xml version="1.0" encoding="gb2312" standalone="no"?>
<!DOCTYPE book[
 <!ELEMENT book (name,author,picture)>
 <!ELEMENT name (#PCDATA)>
 <!ELEMENT author (#PCDATA)>
 <!ELEMENT picture EMPTY>
 <!ATTLIST picture pFile ENTITY #REQUIRED>
 <!ENTITY pEntity SYSTEM "pb.jpg" NDATA jpg>
 <!NOTATION jpg PUBLIC "image/jpeg">
]>
```

```
<book>
 <name>PowerBuilder 数据库应用开发技术</name>
 <author>卢守东</author>
 <picture pFile="pEntity"></picture>
</book>
```

4．内部参数实体

内部参数实体的声明格式为：

```
<!ENTITY % name "value">
```

作为内部参数实体，只能在 DTD 中进行声明，并且只能在 DTD 中被引用。内部参数实体总是包含了可解析的内容。一个到内部参数实体的引用（%name;）将被替换为解析后的内容。

【例 11.25】 内部参数实体的使用示例（11-25.XML）。

```
<?xml version="1.0" encoding="gb2312" standalone="yes"?>
<!DOCTYPE book[
 <!ELEMENT book (name,author)>
 <!ELEMENT name (#PCDATA)>
 <!ENTITY % author "<!ELEMENT author (#PCDATA)>">
 %author;
]>
<book>
 <name>PowerBuilder 数据库应用开发技术</name>
 <author>卢守东</author>
</book>
```

5．外部参数实体

外部参数实体的声明格式为：

```
<!ENTITY % name SYSTEM "systemId">
<!ENTITY % name PUBLIC "publicId" "systemId">
```

除了实体所代表的内容包含在某个外部资源之中以外，外部参数实体的使用方法与内部参数实体的使用方法是完全相同的，也只能在 DTD 中被引用。外部参数实体总是包含了可解析的内容。在 DTD 中，一个到外部参数实体的引用（%name;）将被替换为解析后的内容。

【例 11.26】 内部参数实体的使用示例（11-26.XML）。

```
<?xml version="1.0" encoding="gb2312" standalone="no"?>
<!DOCTYPE book[
 <!ELEMENT book (name,author,date)>
 <!ELEMENT name (#PCDATA)>
 <!ELEMENT author (#PCDATA)>
 <!ENTITY % date SYSTEM "date.dtd">
```

```
 %date;
]>
<book>
 <name>PowerBuilder 数据库应用开发技术</name>
 <author>卢守东</author>
 <date>
  <year>2006</year>
  <month>6</month>
  <day>1</day>
 </date>
</book>
```

其中，外部资源文件 date.dtd 的内容为：

```
<!ELEMENT date (year,month,day)>
<!ENTITY % pcd "(#PCDATA)">
<!ELEMENT year %pcd;>
<!ELEMENT month %pcd;>
<!ELEMENT day %pcd;>
```

11.4.6 DTD 的符号声明

在 DTD 中，符号（Notation）是使用 NOTATION 关键字（大写形式）来进行声明的，其格式为：

```
<!NOTATION name SYSTEM "systemId">
<!NOTATION name PUBLIC "publicId">
<!NOTATION name PUBLIC "publicId" "systemId">
```

其中，name 为符号名称，systemId 与 publicId 为与符号名称相关联的公共标识符或者系统标识符。

通过符号声明，可将指定的符号名称与类型标识符相关联。类型标识符可以是系统标识符，也可以是公共标识符。公共标识符通常是某一种 MIME 类型，而系统标识符则通常是处理相应非解析数据的外部应用程序。

符号的声明与非解析实体的声明是密切相关的。在声明非解析实体时，要指定相应的符号名称。而在声明符号时，再将符号的名称与资源的类型或处理程序相关联。

【例 11.27】 符号声明示例（11-27.XML）。

```
<?xml version="1.0" encoding="gb2312" ?>
<!DOCTYPE author [
 <!ELEMENT author (#PCDATA)>
 <!ATTLIST author photo ENTITY #REQUIRED>
 <!ENTITY PhoEntity SYSTEM "LSD.jpg" NDATA jpg>
 <!NOTATION jpg PUBLIC "image/jpeg">
]>
<author photo="PhoEntity">卢守东</author>
```

11.4.7 DTD 的注释添加

在 DTD 中，同样可以添加相应的注释，其格式为：

```
<!-- 注释的内容 -->
```

实际上，该格式与在 XML 文档中添加注释的格式是一样的。但应该注意的是，注释不能插入到声明之中，而必须放置于声明之外。

如以下示例：

```
<!--声明实体 PhoEntity -->
<!ENTITY PhoEntity SYSTEM "LSD.jpg" NDATA jpg>
<!--声明符号 jpg -->
<!NOTATION jpg PUBLIC "image/jpeg">
```

实 验 11

1. 请自行创建一个 XML 文档，内容为本班同学的有关信息。
2. 请自行创建一个 XML 文档，内容为各学期所开设的课程。
3. 请分别为自己所创建的 XML 文档添加相应的 DTD 声明。

习 题 11

1. 请简述 XML 的基本概念及其主要特点。
2. 请简述 XML 文档的基本结构。
3. 请简述在 XML 中使用标记时所应遵循的基本规则。
4. 请简述 XML 声明的基本格式与使用要点。
5. 请简述 XML 中元素的使用要点。
6. 请简述 XML 中属性的使用要点。
7. 请简述 XML 中注释的使用要点。
8. 请写出 XML 中预定义的五个字符实体及其引用方式。
9. 请简述 XML 中 CDATA 段的使用要点。
10. 请简述 DTD 的基本概念及其主要作用。
11. 在 XML 文档中，如何进行 DTD 的声明？
12. 在 DTD 中，如何进行元素的声明？
13. 在 DTD 中，如何进行属性的声明？
14. 在 DTD 中，如何进行实体的声明？
15. 在 DTD 中，如何进行符号的声明？
16. 在 DTD 中，如何添加相应的注释？

第 12 章　PHP 5 和 XML 文档

本章导读

XML 是 Internet 中用于数据交换的可扩展标记语言，它代替 HTML 是必然的趋势，并且在 Internet 中得到广泛的应用。PHP 提供了多种处理 XML 文档的技术手段，这些技术主要包括 SAX 解析器、DOM 解析器、SimpleXML 解析器、XMLReader 类库以及 XSL 处理函数。前四种技术用来读取和解析 XML 文档，同时完成 XML 的有效性检验。XSL 处理函数用来转换 XML 文档，以便在浏览器上显示其文档内容。本章主要介绍前三种 XML 解析器的使用技术，并通过实例说明它们的应用。

12.1　XML 解析器

XML 解析器是一个用来检查 XML 文档是否格式良好（well-formed）的程序。它通过检查 XML 文档的语法，验证 XML 文档结构是否格式良好。此外，还可以利用文档类型定义（DTD），验证 XML 文档的有效性，XML 解析器根据 DTD 规则，检查 XML 文档中的开始标签和结束标签。

各种主流的 Web 浏览器，如 Mozilla、Konqueror、IE 5 等都包含内置的 XML 解析器，用来解析 XML 文档，以便在浏览器中按指定的格式输出 XML 文档。

XML 解析器为程序员提供二次开发的软件包，程序员在使用 XML 解析器时，调用其提供的函数接口，从而得到解析 XML 的结果。为了使 XML 文档具有更好的可读性和数据显示格式，可以利用 PHP 解析 XML 文档。PHP 3.0 及以上版本都支持 XML 解析。为了解析 XML 文档，需要在 PHP 代码中初始化 XML 解析器。可以在其他应用程序中使用解析后的 XML 文档的数据。根据 XML 解析器的回应调用者的方式不同，可以将 XML 解析器分为基于事件驱动的 XML 解析器和基于树的 XML 解析器两大类。

（1）基于事件驱动的 XML 解析器：这种解析器在内存中不断地处理 XML 数据和 XML 标签，一次处理一个。当 XML 解析器遇到 XML 文档内的任何 XML 元素或数据时，它会产生一个事件，调用一个相应的事件处理程序来完成处理工作。这些事件主要包括标签开始、标签结束、数据处理、注释处理等。例如，SAX 解析器是一个事件驱动的解析器。基于事件驱动的 XML 解析器能够根据 XML 语法规则来检查 XML 文档是否格式良好，但不能作有效性检验。

（2）基于树的 XML 解析器：它一次分析整个 XML 文档，把分析结果保存到一个树状结构中，并提供一组 API 来访问这个树状结构。同时它能够根据由 DTD 定义的规则来检查 XML 文档是否格式良好和有效。

DOM 解析器就是一种基于树的 XML 解析器。DOM 是一个与平台和组织结构无关的模型，DOM 解析器读取 XML 文档，并划分为各种对象，例如元素、属性和注解。DOM 为 XML 文档的每个元素创建一个树状结构，并在内存中保存其树状结构。因此，DOM 解析器与 SAX 解析器相比，可以实现快速搜索 XML 文档的任何元素。

SimpleXML 解析器也是一种基于树的 XML 解析器，它将 XML 文档转换为一个可以用属性和数组来处理的 SimpleXMLElement 对象，把文档的元素看作是对象的属性或者属性数组。通过对 SimpleXMLElement 对象的属性进行操作，进而完成对 XML 文档的相应元素的操作。因此 SimpleXML 是一种简单、易用的 XML 解析器。

12.2 PHP 5 的 SAX 解析器

12.2.1 SAX 解析器的工作原理

SAX（Simple API for XML，XML 简单应用程序接口）解析器是事件驱动的、无有效性检验的 XML 解析器，它用来读取 XML 文档的数据。SAX 的当前版本是 SAX 2。SAX 的工作原理是：SAX 解析器顺序扫描 XML 文档，扫描到元素（element）的开始标签、结束标签等地方时产生事件，由指定的处理函数（handler）完成相应的处理，然后继续扫描其后的内容，直到 XML 文档结束。

SAX 解析器把 XML 文档当成字符串来处理，适合对一种元素重复性地处理，例如，把 XML 文档简单地转换成为 HTML 文档，它只需要把元素替换成对应的 HTML 元素。而且由于它占用很少的内存资源，很适合大型 XML 文档的处理。

当然 SAX 解析器也有其缺点。因为 SAX 解析器没有把整个 XML 文档装入到内存，所以它不能随机读取文档的任何一个元素。此外，SAX 解析器是按顺序处理 XML 文档的，因此 SAX 解析器在处理包含很多内部交叉引用的文档时就会有一些困难，不能实现复杂的搜索。

SAX 解析器检查 XML 文档的结构是否良好，是由各个处理程序来完成的，这些处理程序是在解析 XML 文档，遇到各 XML 标签时被调用。这些处理程序都是用户自定义函数，也称为回调函数。SAX 解析器的工作原理如图 12.1 所示。

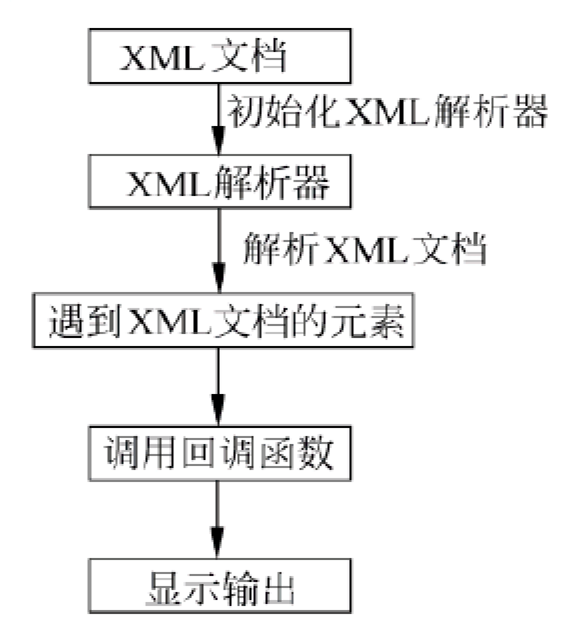


图 12.1　SAX 解析器的体系结构

12.2.2 PHP SAX 解析器的函数库

PHP 的 SAX 解析器是基于 Expat 库的，它在分析 XML 文档时，一般要调用下面的几个函数。

1．创建新的 XML 解析器句柄

函数格式：

```
int xml_parser_create([string encoding])
```

该函数创建一个新的 XML 解析器。它是首先被调用的函数，函数执行成功会返回一个可

被其他 XML 函数使用的 XML 解析器句柄。如果出现错误，则返回 FALSE 值。

在 PHP 5 中，该函数自动检测输入的 XML 的编码，因此，encoding 参数仅用来指定解析后输出数据的编码。解析器支持的编码有 ISO-8859-1、UTF-8 和 US-ASCII。PHP 5.0.2 及以上版本默认输出的字符编码是 UTF-8。

2．注册元素的开始标签和结束标签回调函数

函数格式：

```
bool xml_set_element_handler(int parser,string startElementHandler,string
endElementHandler)
```

该函数用于注册 XML 元素的开始标签和结束标签的回调函数。其中，parser 参数为 XML 解析器句柄，startElementHandler 和 endElementHandler 参数分别为开始标签和结束标签时的回调函数名。注册成功返回 TRUE 值；否则，返回 FALSE。这两个回调函数都是用户自定义函数。

开始标签处理回调函数的格式如下：

```
startElementHandler(int parser, string name, array attribs)
```

当 XML 解析器遇到一个元素的开始标签时，就调用此开始标签回调函数。该函数必须接受三个参数：第 1 个参数 parser 为一个已经创建的 XML 解析器句柄，第 2 个参数 name 为 XML 元素的名称，第 3 个参数 attribs 为一个包含这个元素的所有属性的关联数组。数组元素的下标为属性名称，元素的值为属性的值。关联数组的元素顺序按属性名的顺序排列。

结束标签处理回调函数的格式如下：

```
endElementHandler(int parser, string name)
```

当 XML 解析器遇到一个元素的结束标签时就调用此结束标签回调用函数。该函数必须接受两个参数：第 1 个参数 parser 为一个已经创建的 XML 解析器句柄，第 2 个参数 name 为 XML 元素的名称。

一般先按这两个格式定义好这两个回调函数，它们的函数名是任意的，然后调用 xml_set_element_handler()函数，为 XML 解析器注册这两个回调函数。

3．注册字符数据处理回调函数

函数格式：

```
bool xml_set_character_data_handler(int parser,string handler)
```

该函数用来注册字符数据处理的回调函数。当设置了此回调函数后，XML 解析器扫描到一个元素的内容时就会调用此函数注册的回调函数。注册成功则返回 TRUE 值；否则，返回 FALSE 值。

parser 参数为 XML 解析器句柄，handler 为字符数据处理回调函数的名称，它必须在调用 xml_parse()函数之前定义。

字符数据处理回调函数的格式如下：

```
handler(int parser, string data)
```

函数名称可以是任意的。它必须接受两个参数：第 1 个参数 parser 为指 XML 解析器的句柄，第 2 个参数 data 为 XML 文档中元素的内容，为一个字符串。应该注意的是，对于一段字符内容，回调函数可能会被多次调用。

4．注册指令处理回调函数

函数格式：

```
bool xml_set_processing_instruction_handler(int parser, string handler)
```

该函数用来注册指令处理回调函数，当注册完成后，XML 解析器扫描到指令语句时，就会调用注册的指令处理回调函数，并传入相应的参数。

parser 参数为 XML 解析器句柄，指定的 XML 解析器建立处理指令（PI）处理器函数。handler 参数为指令处理回调函数名。

注册成功则返回 true；否则，返回 false。例如：

```
xml_set_processing_instruction_handler($parser,"xprocess");
```

假设已经定义了 xprocess 函数，上述函数注册一个处理函数 xprocess，当解析器扫描到指令语句时，就调用 xprocess 函数来处理。

指令处理回调函数格式为：

```
handler(int parser,string target,string data )
```

函数名称是任意的，传入的第 1 个参数 parser 是 XML 解析器的句柄；第 2 个参数 target 是处理指令的目标程序，即应用程序名；第 3 个参数是传给目标程序的参数。例如对下面的 XML 指令语句：

```
<?notepad "c:\test.txt" ?>
```

解析器扫描到这一行就调用 xprocess 函数。

```
xprocess($parser,"notepad","c:\test.txt");
```

这样，回调函数可以对指令进行适当的处理并决定是否启动一个进程来运行目标程序，并传入相应的参数。

5．注册注释声明处理回调函数

函数格式：

```
bool xml_set_notation_decl_handler(int parser, string handler)
```

该函数用来为 XML 解析器注册注释声明处理回调函数。当解析器扫描到注释语句时，就会调用注释声明处理回调函数，并传入相应的参数。

parser 参数为 XML 解析器的句柄，handler 是要注册的注释声明处理回调函数名称。

注释声明处理回调函数的原型如下：

```
handler(int parser,string notation_name,string base,string system_id,
string public_id)
```

这个函数需要接受五个参数：第 1 个参数 parser 为 XML 解析器的句柄；第 2 个参数 notation_name 为注释声明定义中的注释名称；第 3 个参数 base 通常设置为空字符串；第 4 个参数 system_id 为外部注释声明的系统标识符；第 5 个参数 public_id 为外部注释声明的公共标识符。例如下面的注释声明语句：

```
<!NOTATION gif SYSTEM "file:///usr/bin/netscape">
```

假设注释声明回调函数名 xnotation，当解析器扫描到这一行就会调用 xnotation 函数，具体的代码如下：

```
xnotation($parser,"gif",""," file:///usr/bin/netscape","");
```

6．注册默认回调函数

函数格式：

```
bool xml_set_default_handler(int parser, string handler)
```

默认回调函数在解析器扫描到一个节点却没有对应的处理函数被注册时调用。例如，如果没有注册开始标签和结束标签处理回调函数，当解析器扫描到一个元素开始标签或者结束标签时，就会试图调用默认回调函数。当然，如果注册了开始标签或者结束标签的处理回调函数，扫描到一个元素开始标签或是结束标签时就不会调用默认回调函数。parser 参数为 XML 解析器的句柄，handler 参数为默认回调函数名称。

默认回调函数的格式为：

```
default_handler(int parser, string data)
```

其中，data 参数为扫描到的无法处理的全部数据，其内容可以是 XML 声明、文档类型声明、实体名或者其他没有已存在处理器的数据。

7．解析 XML 文档的函数

函数格式：

```
int xml_parse(int parser, string data[,bool isFinal])
```

该函数用创建的 XML 解析器开始解析一个 XML 文档。其中，parser 参数为 XML 解析器句柄，data 参数为需要解析的数据集。可以通过改变 data 参数的内容，多次调用这个函数来处理 XML 文档。isFinal 参数是可选的，如果将其设为 true 值，则是最后一次调用 xml_parse()函数，此时，data 的值为当前解析中的最后一段数据。如果成功，则返回 TRUE；失败，则返回 FALSE。

8．释放 XML 解析器的函数

函数格式：

```
bool xml_parser_free(int parser)
```

该函数释放指定的 XML 解析器，parser 参数为要释放的 XML 解析器的句柄，该函数在解析完 XML 文档时使用。

9．获取 XML 解析器错误代码

函数格式：

```
int xml_get_error_code(resource parser)
```

该函数获取 XML 解析器在处理 XML 时的错误代码，parser 参数为 XML 解析器的句柄。如果 parser 参数没有指向一个合法的 XML 解析器，则返回 FALSE 值；否则，将返回错误代码（如 XML_ERROR_NO_ELEMENTS、XML_ERROR_INVALID_TOKEN 等）。

10．获取 XML 解析器错误信息

函数格式：

```
string xml_error_string(int code)
```

该函数根据给定的错误代码 code 值，返回 XML 解析器的错误字符串。code 参数是由 xml_get_error_code()返回的错误代码。返回值是与 code 参数描述的错误代码对应的错误描述信息。

12.2.3 用 PHP 的 SAX 解析器处理 XML 文档

利用 PHP 的 SAX 解析器来分析 XML 文档，关键是为 XML 文档的各元素编写事件处理程序，即回调函数，然后注册这些回调函数。这样 SAX 解析器在扫描 XML 文档时，遇到元素标签、字符数据、处理指令和注释声明时，就自动地调用相应的回调函数来处理。

一般用 SAX 解析 XML 文档的编程思路如下：

（1）为处理 XML 文档的各个回调函数编写程序代码，如定义元素的开始标签和结束标签回调函数、字符数据处理回调函数。

（2）利用 PHP 函数 xml_parser_create()初始化 SAX 解析器。其代码为：

```
$xparser=xml_parser_create();
```

（3）设置各事件调用的回调函数，例如，代码如下：

```
xml_set_element_handler($xparser, "startHandler","endHandler");
xml_set_character_data_handler($xparser, "cdataHandler");
```

代码中，xml_set_element_handler()函数为 XML 文档中元素的开始标签和结束标签设置回调函数，分别是 startHandler 函数、endHandler 函数。xml_set_character_data_handler()函数指定 XML 文档中元素内的字符数据处理回调函数为 cdataHandler。这三个回调函数名是第（1）步定义的函数名。

（4）用 fopen()函数打开 XML 文档，代码如下：

```
if (!($fp=fopen("student.xml","r"))) {
die("文件 student.xml 不存在");
}
```

执行该语句后，建立一个文件指针变量$fp，指向打开的 student.xml 文件。如果 student.xml 文件不存在，则显示一个错误信息，结束程序的执行。

（5）用 xml_parse()函数解析打开的 XML 文档。代码为：

```
while ($data=fread($fp, 4096)) {
   if(!xml_parse($xparser,$data,feof($fp))) {
     die("解析错误: xml_error_string(xml_get_error_code($xparser))");
   }
}
```

上述代码循环读取 XML 文档内容，每次循环读取 4KB 的内容，由 xml_parse()函数解析 XML 文档直到 XML 文档末尾。当到达 XML 文档末尾时，feof()函数返回 true 值，并通知解析器结束处理。

（6）利用 xml_parser_free()函数释放 XML 解析器的句柄。代码为：

```
xml_parser_free($xml_parser);
```

下面通过几个例子来说明 SAX 解析器的应用。

【例 12.1】 建立一个格式良好的 XML 文档，存放了两个学生的信息，包括姓名、学号、年龄、地址和所在系，文件名为 students.xml。

```
<?xml version="1.0" encoding="UTF-8"?>
<STUDENTDETAIL>
    <STUDENT>
        <NAME ID="S001">张大全</NAME>
        <AGE>20</AGE>
        <ADDRESS>广州市</ADDRESS>
        <DEPARTMENT>计算机系</DEPARTMENT>
    </STUDENT>
    <STUDENT>
        <NAME ID="S002">黄浩</NAME>
        <AGE>21</AGE>
        <ADDRESS>上海市</ADDRESS>
        <DEPARTMENT>会计系</DEPARTMENT>
    </STUDENT>
</STUDENTDETAIL>
```

接下来编写 PHP 程序，利用 SAX 解析器来分析该 XML 文档。

【例 12.2】 用 SAX 解析器分析 students.xml 文件的 PHP 程序（ex12_2.php）。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>SAX 解析器示例</title>
</head>
<body>
<?php
function startHandler($xparser, $element_name, $attributes) {
```

```
        echo "开始标签:<b>$element_name</b><br>";
        while (list($key,$value)=each($attributes)) {
            echo "属性:<b><i>$key=$value</i></b><br>";
        }
    }
    function endHandler($xparser, $element_name) {
        echo "结束标签:<b>$element_name</b><br>";
    }
    function cdataHandler($xparser, $cdata) {
        echo "字符数据: <i><u>$cdata</u></i><br>";
    }
    //主程序
    $xfile="students.xml";
    $xparser=xml_parser_create("UTF-8");
    xml_set_element_handler($xparser, "startHandler","endHandler");
    xml_set_character_data_handler($xparser,"cdataHandler");
    if(!($fp=fopen($xfile,"r"))) {
        die("文件打开错误: $xfile");
    }
    while($data=fread($fp,4096)) {
        if(!xml_parse($xparser,$data,feof($fp))) {
            $err=xml_get_error_code($xparser);
            die("XML 解析器错误: xml_error_string($err)");
        }
    }
    xml_parser_free($xparser);
    ?>
    </body>
    </html>
```

在上述程序代码中，定义了 startHandler、endHandler 和 cdataHandler 三个用户函数，它们分别作为开始标签处理函数、结束标签处理函数和元素标签内的字符数据处理函数。通过 xml_set_element_handler()和 xml_set_character_data_handler()函数将它们注册为 XML 解析器的回调函数。当解析 XML 文档，扫描到元素的开始标签时，就调用 startHandle()函数，以粗体形式显示开始标签名。PHP 的 list()和 each()函数访问数组变量，解析器也以斜体、粗体形式显示元素标签的属性名和属性值。当解析器扫描到结束标签时调用 endHandler()函数，以粗体形式显示 XML 文档结束标签的名字。当解析器扫描到开始标签和结束标签之间的文本时，调用 cdataHandler()函数，以下划线和斜体格式显示开始标签和结束标签之间的文本。例 12.2 的输出结果如图 12.2 所示。

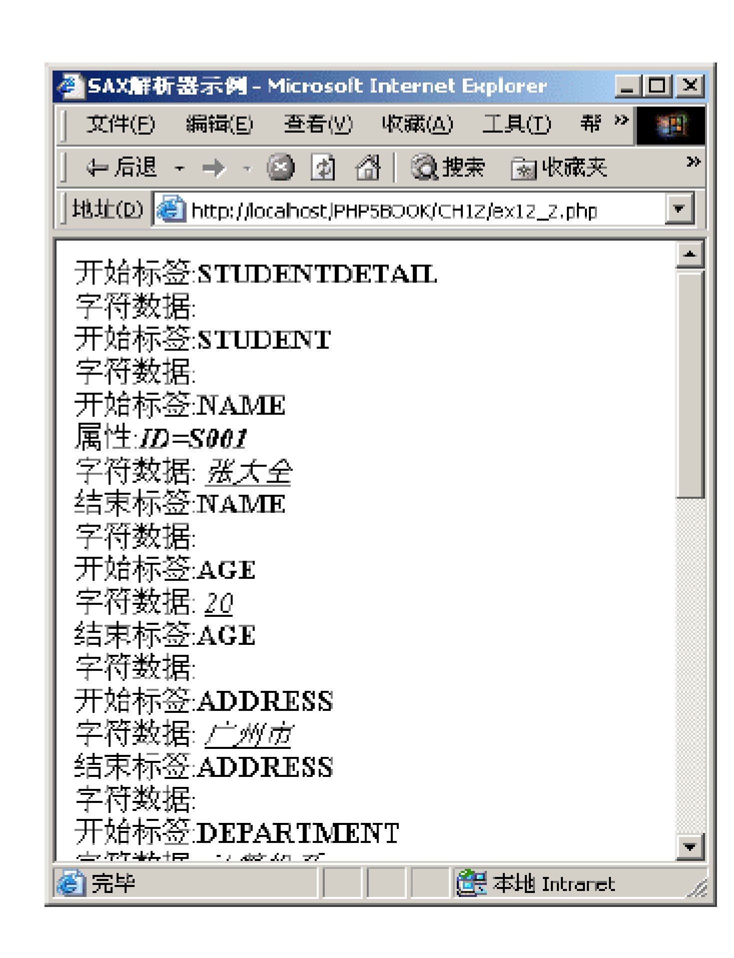


图 12.2　例 12.2 的输出结果

【例 12.3】 用 SAX 将 XML 文档转换为 HTML 表格的 PHP 程序，文件名为 ex12_3.php。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>SAX 解析器示例</title>
</head>
<body>
<?php
function startHandler($xparser, $element_name, $attributes) {
   if ($element_name=="STUDENTDETAIL")
       echo '<table border="1" cellpadding="0" cellspacing="0">';
   else if ($element_name=="STUDENT")
        echo "<tr>";
   else if ($element_name=="NAME")
    {  list($key,$value)=each($attributes);
      echo "<td>".$value."</td><td>";
   }
   else
        echo "<td>";
}
function endHandler($xparser, $element_name) {
   if ($element_name=="STUDENTDETAIL")
       echo "</table>";
   else if ($element_name=="STUDENT")
        echo "</tr>";
   else
        echo "</td>";
}
function cdataHandler($xparser, $cdata) {
   echo $cdata;
}

$xfile="students.xml";
$xparser=xml_parser_create("UTF-8");
xml_set_element_handler($xparser, "startHandler","endHandler");
xml_set_character_data_handler($xparser,"cdataHandler");
if(!($fp=fopen($xfile,"r"))) {
   die("文件打开错误: $xfile");
}
while($data=fread($fp,4096)) {
   if(!xml_parse($xparser,$data,feof($fp))) {
      $err=xml_get_error_code($xparser);
      die("XML 解析器错误: xml_error_string($err)");
```

```
        }
    }
    xml_parser_free($xparser);
    ?>
    </body>
    </html>
```

本例的程序代码利用 SAX 解析器，将例 12.1 的 students.xml 文件的 XML 内容转换为 HTML 表格的形式，输出到浏览器。其转换方法实际上是将 XML 文档的元素转换为 HTML 表格的相应标记，例如，在开始标签处理函数 startHandler 中，XML 的 STUDENTDETAIL 元素转换为<table>标记，STUDENT 元素转换为<tr>标记，NAME 元素含有属性 ID，在转换时要在单元格中输出 ID 属性的值和姓名，因此作特殊的处理，表示一个学生信息的元素转换为<td>标记。类似地，在结束标签处理函数 endHandler 中，XML 的 STUDENTDETAIL 元素转换为</table>标记，STUDENT 元素转换为</tr>标记，表示一个学生信息的各个元素转换为<td>标记。该程序的输出结果如图 12.3 所示。

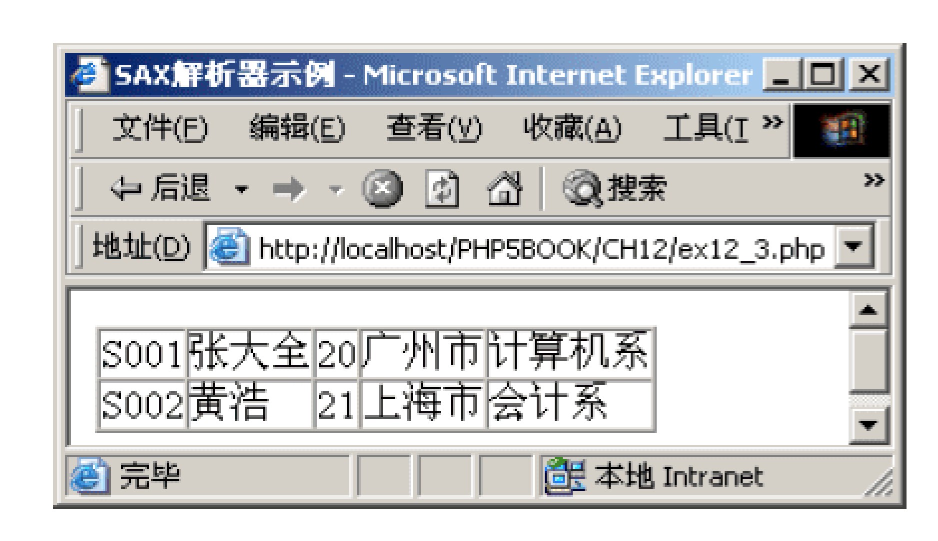


图 12.3　例 12.3 的输出结果

12.3　PHP 5 的 DOM 解析器

12.3.1　DOM 概述

DOM（document object model）即文档对象模型。DOM 是一组创建和编辑 XML 文档的标准编程接口。编程者可以使用 DOM 动态地创建 XML 文档，遍历文档结构，添加、修改、删除 XML 文档的元素，改变文档的显示方式等。W3C 组织把 DOM 分为三个不同级别的规范：1 级、2 级和 3 级，每个级别又增加了一些新的函数。PHP 5 实现了 DOM 的 1 级和 2 级规范。

DOM 将一个 XML 文档以对象树的形式来表示，该对象树包含若干个节点，这些节点之间存在父子关系。每个节点是一个对象，代表 XML 文档的元素、属性、注释等。每个对象封装了操作 DOM 树的方法和属性。此后对该 XML 文档的所有操作（如插入、查询元素），均转化为对该对象树的操作。例如，下面的 emp.xml 文件存放员工的信息，包括员工编号、姓名、参加工作日期。其内容如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<Company>
    <Employee ID="1001">
        <Name>徐华</Name>
        <DateOfHiring>1990-12-20</DateOfHiring>
```

```
    </Employee>
<!-- 注释行 -->
</Company>
```


一旦把该 XML 文档装入到 DOM 解析器，就通过 DOM 对象来表示 XML 文档的结构，如图 12.4 所示。

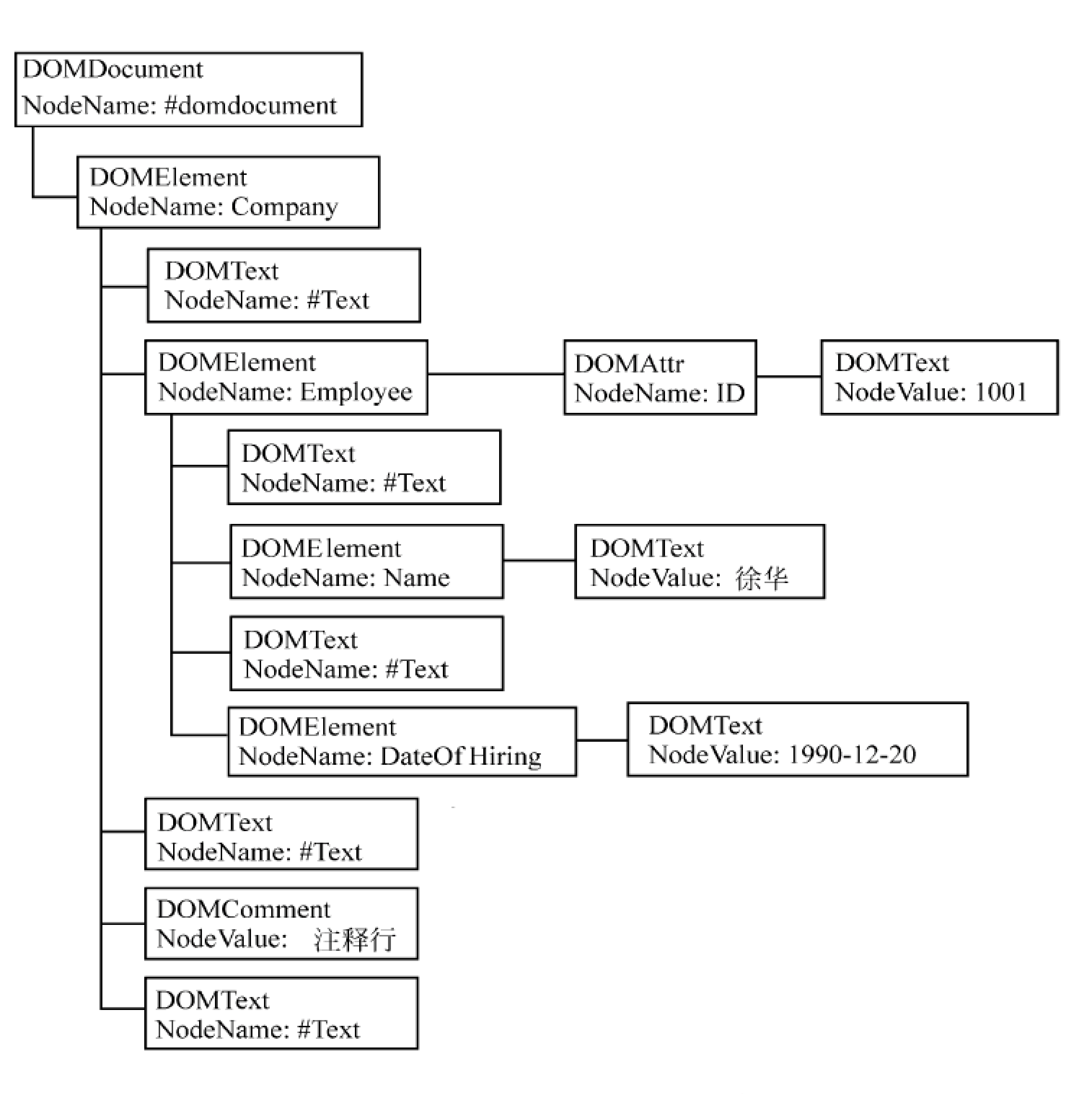


图 12.4 emp.xml 的 DOM 树结构

说明：在默认情况下，PHP 5 的 DOM 装入 XML 文档时，将换行符、空格、Tab 符等空白符作为文本节点，包括在 DOM 树结构中。这在遍历 XML 文档内容时要去掉这些附加的文本节点。

DOM 使用基于树的方法来解析 XML 文档，DOM 将整个 XML 文档读取到内存，转换为树结构，从而可以方便地进行查找、插入和删除元素等操作。不过，DOM 需要占用大量的内存，适合处理较小的 XML 文档。

DOM 提供了各种内置的类，用于在 PHP 5 中解析 XML 文档。常用的类有 DOMDocument 类、DOMNode 类、DOMElement 类、DOMText 类及 DOMAttr 类等。PHP 5 的 DOM 解析器通过建立这些类的对象实例，描述 XML 文档。下面介绍这些类的基本用法，然后给出几个操作实例。

12.3.2 DOMDocument 类

DOMNode 类是 DOM API 中的大多数类的基类，提供了几个公共函数，不能作为单独的类来使用。DOMDocument 类派生于 DOMNode 基类，用来创建、装入和保存 XML 文档对象实例，以及提供创建其他节点类型对象的方法。

1．创建文档对象

当使用 DOM 处理文档时，第一步是用 DOMDocument 类创建或者装入一个文档对象，表示整个 XML 文档的树结构。DOMDocument 类的构造方法如下：

```
DOMDocument([string version], [string encoding])
```

version 参数和 encoding 参数是可选的，分别表示 XML 文档的版本号、文档的字符编码。可以使用 new 关键字初始化一个空的文档对象。

```
$dom = new DOMDocument('1.0');
```

它创建一个空的 DOMDocument 对象$dom，使用 XML 1.0 规范，不指定字符编码。它等价于下面的 XML 声明：

```
<?xml version="1.0"?>
```

version 参数默认值是 1.0，因此，在初始化对象实例时可以省略此参数。当使用 encoding 参数时，必须给出 version 参数值。例如，以下代码

```
$dom = new DOMDocument('1.0', 'ISO-8859-1');
```

将产生一个 XML 声明：<?xml version="1.0" encoding="ISO-8859-1"?>。

上述两个例子创建的$dom 对象变量都是用 DOMDocument 类的构造方法来初始化为一个空的文档对象。通过$dom 对象，可以用 DOM API 来创建一个对象树，或者装入一个 XML 文档来创建一个对象树。

2．装入 XML 文档

使用$dom 对象，可以用 loadXML()方法装入一个 XML 字符串或者用 load()方法装入一个 XML 文档来创建一个 DOM 对象树。这两个方法的语法如下：

```
loadXML(string source [, int options])
load(string filename [, int options])
```

在 loadXML()方法中，source 参数为一个包含 XML 文档的字符串。Load()方法的 filename 参数为一个指向 XML 文件的 URI。当使用 PHP 5.1 及以上版本时，这两个方法的第 2 个参数指定 DOM 解析器在生成对象树时的选项常量，有关这些选项常量的内容参见 PHP 5 的 libxml 函数部分。例如：

```
$xmldata = '<?xml version="1.0"?>
<root>
<child>contents</child>
```

```
</root>';
$dom->loadXML($xmldata, LIBXML_NOBLANKS);
```

$xmldata 字符串变量存放一个需要装入的 XML 文档，在 loadXML()方法指定 LIBXML_NOBLANKS 常量，表示删除对象树中的所有空白符，这些空白符是无用的。该代码段与下面代码段的作用相同。

```
$xmldata = '<?xml version="1.0"?><root><child>contents</child></root>';
$dom->loadXML($xmldata);
```

在 load()方法中，filename 参数值可以是本服务器上的文件或者远程主机的文件。例如：

```
/* xmldata.xml 文件与脚本在同一个 Web 目录中 */
$dom->load('xmldata.xml', LIBXML_NOBLANKS);
/* 用绝对路径指定文件的位置 */
$dom->load('file:///tmp/xmldata.xml', LIBXML_NOBLANKS);
/* 从远程主机装入 xmldata.xml 文件 */
$dom->load('http://www.example.com/xmldata.xml', LIBXML_NOBLANKS);
```

除了上面介绍的初始化 DOMDocument 对象来装入 XML 文档，生成对象树的方法外，也可以静态方式来调用 DOMDocument 类的 load()方法，将 XML 数据装入到树中，同时返回新建的 DOMDocument 对象。例如，下面代码生成的$dom 对象与前面的相同。为简单起见，删除了 XML 声明。

```
/* 从字符串装入到树*/
$dom = DOMDocument::loadXML('<root><child>contents</child></root>');
/* 从 URI 装入到树*/
$dom = DOMDocument::load('xmldata.xml', LIBXML_NOBLANKS);
```

3．删除文档的空白符

在 PHP 5 中，通过设置 DOMDocument 对象的 preserveWhiteSpace 属性值为 TRUE，则保留文档中的所有空白符，如果将其值设置为 FALSE，则删除文档的空白符。当使用这一属性时，必须在调用 load()函数之前设置该属性的值。例如，下面的代码与带有 LIBXML_NOBLANKS 选项的 load()方法是等价的。

```
$dom = new DOMDocument();
$dom->preserveWhiteSpace = FALSE;  //删除文档的空白符号
$dom->load('xmldata.xml');
```

当同时使用 preserveWhiteSpace 属性和解析器选项时，解析器选项优先于该属性。即在任何设置了该属性的 DOMDocument 对象实例中，只传递与属性冲突的解析器选项，DOM 解析器将执行解析器选项指令，例如：

```
$dom = new DOMDocument();
$dom->preserveWhiteSpace = TRUE;    //保留文档中的空白符
$dom->load('xmldata.xml', LIBXML_NOBLANKS);
```

在这个例子中，因 LIBXML_NOBLANKS 选项优先于 preserveWhiteSpace 属性，因此装入文档时删除文档的空白符。

4．输出 DOMDocument 对象内容

DOMDocument 对象的 saveXML()和 save()方法用来输出 XML 文档内容。这两个方法的语法如下：

```
saveXML([DOMNode node [, integer options]])
save(string filename [, integer options])
```

saveXML()方法将 DOM 对象树或者其指定节点的内容输出到一个字符串。执行成功时，返回值为一个 XML 字符串；否则，返回 FALSE 值。

在 saveXML()方法中，可选的 node 参数为节点参数，用来输出指定节点的内容。它必须是由 DOMNode 类派生的对象，而且必须是与 DOMDocument 对象相同的文档中派生的对象。当没有 node 节点参数时，将整个文档对象转换为一个字符串。

【例 12.4】 用 DOM 装入 XML 文档，然后输出 XML 文档（ex12_4.php）。

```
<?
$xmldata = '<?xml version="1.0" encoding="GB2312"?><root><child>子节点内容
</child></root>';
$dom = new DOMDocument('1.0');
$dom->loadXML($xmldata);
print $dom->saveXML();
?>
```

输出结果为一个字符串，如下：

```
<?xml version="1.0" encoding="GB2312"?>
<root><child>子节点内容</child></root>
```

save()方法将 DOM 对象树内容输出到一个由 filename 参数指定的文件中。返回值是保存到文件的字节数。它与 saveXML()方法不同，不能保存单个节点内容到文件中。例如，将上述例 12.4 的 print 语句改为以下语句。

```
$bytes = $dom->save('output.xml);
```

这一语句将 XML 文档保存到 output.xml 文件，返回被保存的字节数，保存到$bytes 变量。

5．格式化文档

一般情况下，用 saveXML()或 save()方法输出 XML 文档时，结果是所有节点元素排列在一起的字符串，不易于阅读。输出之前，将 DOMDocument 对象的 formatProperty 属性设置为 TRUE，可以美化文档的输出格式，使 DOM 解析器在文档中增加换行符，并向右缩进。

【例 12.5】 格式化文档（ex12_5.php）。

```
<?
$xmldata = '<?xml version="1.0" encoding="GB2312"?><root><child>子节点内容
</child></root>';
$dom = new DOMDocument('1.0');
$dom->loadXML($xmldata);
$dom->formatOutput = TRUE;
print $dom->saveXML();
?>
```

输出结果如下：

```
<?xml version="1.0" encoding="GB2312"?>
<root>
  <child>子节点内容</child>
</root>
```

6．读取 XML 文档的根元素

根元素是 XML 文档体的顶级元素，它是一个节点，通过访问 DOMNode 类派生的对象的 documentElement 属性，可以读取文档的根元素，返回值为 DOMElement 对象。例 12.5 的根元素是 root 元素，使用 DOMDocument 对象变量$dom 的 documentElement 属性来检索 root 元素：

```
$root = $dom->documentElement;
```

这个语句返回一个 DOMElement 对象，它是 root 元素节点，并保存到$root 变量。

7．创建元素节点对象

利用 DOMDocument 对象的 createElement()方法，可以创建与 XML 文档关联的新元素节点。createElement()方法的语法为：

```
createElement(string name[, string value])
```

该方法创建一个新节点。第 1 个参数是要创建的元素名称，第 2 个可选参数指定元素的值。返回值是一个 DOMElement 对象。

【例 12.6】 创建 XML 文档的元素节点（ex12_6.php）。

```
<?php
$dom = new DOMDocument('1.0', 'UTF-8');
$dom->formatOutput = TRUE;
$root = $dom->createElement('root');
$dom->appendChild($root);        // 将 root 元素插入到树中，作为根元素
$child = $dom->createElement('child', '这是子节点元素');
$root->appendChild($child);      //将 child 元素添加到 root 元素的子节点
echo $dom->saveXML();
?>
```

输出结果为：

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <child>这是子节点元素</child>
</root>
```

注意：用 createElement()方法创建的新元素只与节点对象关联，但不会出现在文档中。只有调用 DOMNode 类的 appendChild()方法，才能将新元素节点插入到对应的 XML 文档中。appendChild()需要带一个要添加的节点参数，将节点参数指定的节点添加到某一节点的子节点中。

8．创建属性节点对象

DOMDocument 对象的 createAttribute()方法用来创建属性节点，其语法如下：

```
createAttribute(string name)
```

该方法的返回值为一个 DOMAttr 对象，表示新创建的属性名。name 参数是要创建的属性名。

使用这一方法，还需要用 DOMNode 类的 nodeValue 属性或者用 DOMAttr 类的 value 属性来指定新建属性的值。此外，还需要调用 DOMNode 类的 appendChild()方法，将新建的属性添加到某一节点对象中。

【例 12.7】 创建元素的属性（ex12_7.php）。

```
<?php
$dom = new DOMDocument('1.0', 'UTF-8');
$dom->formatOutput = TRUE;
$root = $dom->createElement('root','根元素');
$dom->appendChild($root);
$attr = $dom->createAttribute("id");
$attr->nodeValue = "1001";      //与$attr->Value = "1001";作用相同
$root->appendChild($attr);      //将 id 属性添加到 root 元素
echo $dom->saveXML();
?>
```

输出结果如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<root id="1001">根元素</root>
```

12.3.3 DOMNode 类

DOMNode 类是 DOM 扩展的基类，它提供的方法和属性可用于派生于该类的子类。

1．DOMNode 类的属性

DOMNode 类的属性如表 12.1 所示。

表 12.1　DOMNode 类的属性

属性名	值类型	说明
nodeName	string	当前节点名字，文档节点的名字为#document，文本节点的名字为#text
nodeValue	string	当前节点的值，其值可更改。文档节点的值为 NULL，输出一个换行符
nodeType	int	节点类型，其值见表 12.2
parentNode	DOMNode	当前节点的父节点
childNodes	DOMNodeList	当前节点的子节点集合。如果没有子节点，则返回空的 DOMNodeList 集
firstChild	DOMNode	当前节点的第一个子节点。如果没有子节点，返回 NULL
lastChild	DOMNode	当前节点的最后一个子节点。如果没有子节点，返回 NULL
previousSibling	DOMNode	当前节点的上一个兄弟节点。如果没有这节点，返回 NULL
nextSibling	DOMNode	当前节点的下一个兄弟节点。如果没有这节点，返回 NULL
attributes	DOMNamedNodeMap	当前节点的属性集，为 DOMNamedNodeMap 类型
ownerDocument	DOMDocument	与当前节点关联的文档，返回 DOMDocument 对象
textContent	string	返回当前节点及其子孙的文本内容

节点类型常量如表 12.2 所示。

表 12.2　节点类型常量

常量名	值	说明
XML_ELEMENT_NODE	1	节点是 DOMElement 对象
XML_ATTRIBUTE_NODE	2	节点是 DOMAttr 对象
XML_TEXT_NODE	3	节点是 DOMText 对象
XML_CDATA_SECTION_NODE	4	节点是 DOMCharacterData 对象
XML_ENTITY_REF_NODE	5	节点是 DOMEntityReference 对象
XML_ENTITY_NODE	6	节点是 DOMEntity 对象
XML_PI_NODE	7	节点是 DOMProcessingInstruction 对象
XML_COMMENT_NODE	8	节点是 DOMComment 对象
XML_DOCUMENT_NODE	9	节点是 DOMDocument 对象
XML_DOCUMENT_TYPE_NODE	10	节点是 DOMDocumentType 对象

【例 12.8】 输出 xmp.xml 文件的元素名（ex12_8.php）。

```
<?
$dom = new DOMDocument();
$dom->load("emp.xml");
$root=$dom->documentElement;
print "文档节点名: ".$dom->nodeName."\n";
print "根元素节点名: ".$root->nodeName."\n";
if ($root->hasChildNodes()) {
   $children = $root->childNodes;
```

```
   foreach($children as $node)
      print $node->nodeName."\n";
}
?>
```

该程序输出文档节点名、根元素节点名和根元素的下一级子节点名。用 foreach 循环语句遍历 DOMNodeList 集合，输出每个节点的名字。输出结果如下：

```
文档节点名：#document
根元素节点名：Company
#text
Employee
#text
#comment
#text
```

由于 emp.xml 文件中的 Company 元素包含两个子元素，它们之间用换行符分开。装入文档时，这些换行符未被删除，从而在 DOM 对象树中将这两个换行符创建为文本节点。因此，输出两个#text。#comment 表示注释节点名。为了删除这些多余的文本节点，可以将 load()方法改为以下语句：

```
$dom->load("emp.xml",LIBXML_NOBLANKS);
```

或者将 foreach 循环语句改为以下语句：

```
foreach($children as $node)
  if ($node->nodeType != XML_TEXT_NODE) {
     print $node->nodeName."\n";
  }
```

【例 12.9】 输出 xmp.xml 文件的 Company 根元素的值（ex12_9.php）。

```
<?
$dom = new DOMDocument();
$dom->load("emp.xml",LIBXML_NOBLANKS);
$root=$dom->documentElement;
print $root->nodeValue;
?>
```

$root->nodeValue 返回的值是 Company 根元素下所有文本节点的值，输出结果为一个字符串：徐华 1990-12-20。

2．DOMNode 类的方法

1）appendChild()

格式：DOMNode appendChild(DOMNode newnode)

给当前节点增加一个子节点。子节点由 DOMDocument 对象的 createElement()、createTextNode()等方法来创建。newnode 参数是要增加的子节点对象。返回值是新增的节点对象。

2）cloneNode()

格式：DOMNode cloneNode([bool deep])

复制当前节点。deep 参数表示是否复制当前节点的所有子孙节点，默认值为 FALSE，返回值为复制的节点。

3）hasAttributes()

格式：bool hasAttributes(void)

检测当前节点是否有属性。被检测的节点必须是 XML_ELEMENT_NODE 类型的节点，即元素节点。如果有属性则返回 TRUE；否则，返回 FALSE。

4）hasChildNodes()

格式：bool hasChildNodes(void)

检测当前节点是否有子节点。如果有子节点则返回 TRUE；否则，返回 FALSE。

5）insertBefore()

格式：DOMNode insertBefore(DOMNode newnode [, DOMNode refnode])

在指定节点前插入一个新节点。如果要进一步修改插入的节点，必须使用返回的节点。

newnode 参数是新增的节点。refnode 参数是已存在的参考节点，如果未指定其值，则新节点 newnode 插入到当前节点的子节点中。返回值为插入的节点。

6）isSameNode()

格式：bool isSameNode(DOMNode node)

检测当前节点与 node 参数指定的节点是否为同一个节点。如果两者是同一节点，则返回 TRUE；否则，返回 FALSE。

例如，以下代码的输出结果为 TRUE，表明$node 变量和$dom 变量的值都是同一个文档节点。

```
<?
$dom = new DOMDocument();
$dom->load("emp.xml");
$root=$dom->documentElement;
$node=$root->ownerDocument;
if ($dom->isSameNode($node))
   print "TRUE";
?>
```

7）removeChild()

格式：DOMNode removeChild(DOMNode oldnode)

删除当前节点的子节点。oldnode 参数是要删除的子节点。如果子节点被删除，则返回原来的节点。

8）replaceChild()

格式：DOMNode replaceChild(DOMNode newnode, DOMNode oldnode)

用新节点替换原节点。newnode 参数是新节点，oldnode 参数是原节点。如果替换成功，则返回原节点；否则，返回 FALSE。

12.3.4 DOMElement 类

DOMElement 类派生于 DOMNode 基类，用来定义 XML 文档的元素，一个 DOMElement 对象代表 XML 文档的一个元素。

1．DOMElement 类的构造方法

格式：DOMElement(string name[,string value[,string namespaceURI]])

该构造方法创建一个新的 DOMElement 对象，表示一个元素节点。此对象是只读的，可以添加到文档中，但是不能将其他节点添加到 DOMElement 对象所指的节点，除非该节点与文档关联。

name 参数指定元素标签名，value 参数指定元素的值，namespaceURI 参数指定名称空间 URI，用来在指定名称空间内创建元素。

【例 12.10】 创建元素节点的简单示例（ex12_10.php）。

```
<?php
$dom = new DOMDocument('1.0', 'iso-8859-1');
$element = $dom->appendChild(new DOMElement('root'));
$element_ns = new DOMElement('pr:node1', 'thisvalue', 'http://xyz');
$element->appendChild($element_ns);
echo $dom->saveXML();
?>
```

结果为：

```
<?xml version="1.0" encoding="iso-8859-1"?>
<root><pr:node1 xmlns:pr="http://xyz">thisvalue</pr:node1></root>
```

2．DOMElement 类的属性

DOMElement 类的属性只有一个 tagName 属性，存放元素名。

3．DOMElement 类的方法

1）getAttribute()

格式：string getAttribute(string name)

读取当前节点中由 name 参数指定的属性值。name 参数为属性名，返回值为属性的值，如果指定的属性不存在，则返回空字符串。

2）getAttributeNode()

格式：DOMAttr getAttributeNode(string name)

返回当前元素节点中指定属性名的属性节点，返回值为 DOMAttr 对象。

3）getElementsByTagName()

格式：DOMNodeList getElementsByTagName(string name)

返回当前元素节点下由 name 参数指定的所有子孙元素组成的一个 DOMNodeList 对象。name 参数为元素标签名，可以用“*”代表指定元素树和所有元素。

【例 12.11】 输出例 12.1 的 students.xml 文件中的所有学生名单（ex12_11.php）。

```
<?
$dom = new DOMDocument();
$dom->load("students.xml");
$namelist = $dom->getElementsByTagName("NAME");
$length = $namelist->length;
for ($x=0;$x < $length;$x++) {
    print "姓名: ".$namelist->item($x)->nodeValue."\n";
}
?>
```

该程序用 getElementsByTagName()方法检索由$dom 所指文档内的所有 NAME 元素，存放到$namelist 变量。$namelist 是一个集合，根据其 length 属性值（集合元素的个数），利用 for 循环遍历该集合，输出其集合元素的 nodeValue 值。结果如下：

```
姓名: 张大全
姓名: 黄浩
```

4）hasAttribute()

格式：bool hasAttribute(string name)

测试当前元素节点是否存在由 name 参数指定的属性。name 参数为属性名。如果存在该属性，则返回 TRUE；否则，返回 FALSE。

5）removeAttribute()

格式：bool removeAttribute(string name)

删除当前元素节点中由 name 参数指定的属性，name 参数为属性名。如果删除成功，则返回 TRUE；失败，则返回 FALSE。

6）removeAttributeNode()

格式：bool removeAttributeNode(DOMAttr oldnode)

删除当前元素节点的指定属性。oldnode 参数是要删除的属性节点对象。如果删除成功，则返回 TRUE；失败，则返回 FALSE。

7）setAttribute()

格式：bool setAttribute(string name, string value)

在当前元素节点中增加一个属性及其值。如果属性不存在，则创建该属性。name 参数是属性名，value 是属性的值。

8）setAttributeNode()

格式：DOMAttr setAttributeNode(DOMAttr attr)

在当前元素节点中增加一个新的属性节点。attr 参数是新增的属性节点对象。如果属性被替换，返回原属性节点；否则，返回 NULL。

【例 12.12】 修改 emp.xml 文件的第一个员工信息：增加一个 deptID 属性，值为 01；在 DateOfHiring 元素前插入一个 Age 元素（ex12_12.php）。

```
<?
$xmlfile="emp.xml";
$dom = new DOMDocument();
$dom->load($xmlfile,LIBXML_NOBLANKS);
$root = $dom->documentElement;
$employee=$root->childNodes->item(0);        //第 1 个 Employee 元素节点
$employee->setAttribute("deptID", "01");    //给 Employee 元素增加属性 deptID
$DateHire=$employee->getElementsByTagName("DateOfHiring")->item(0);
$Age=new DOMElement('Age',25);               //创建一个 Age 元素对象
$employee->insertBefore($Age,$DateHire);//在 DateOfHiring 元素前插入 Age 元素
$dom->save($xmlfile);
?>
```

12.3.5 DOMText 类

文本节点是简单节点，只包含文本内容，不含有子节点和属性。利用 DOMText 类，可以表示 XML 文档的字符数据。可以用 DOMDocument 对象的 createTextNode()方法或者用 new 关键字实例化 DOMText 对象来创建文本节点，然后用 appendChild()和 insertBefore()方法插入文本节点到 DOM 树中。

1．DOMText 类的构造方法

格式：DOMText([string value])

创建一个 DOMText 对象。value 参数为文本节点的值，如果未指定其值，则创建一个空的文本节点。

【例 12.13】 创建和插入文本节点（ex12_13.php）。

```
<?php
$dom = new DOMDocument('1.0', 'UTF-8');
$root = $dom->appendChild(new DOMElement('book'));
$title = $root->appendChild(new DOMElement('booktitle'));
$text = $title->appendChild(new DOMText('C++程序设计教程'));
echo $dom->saveXML();
?>
```

输出结果为：

```
<?xml version="1.0" encoding="UTF-8"?>
<book><booktitle>C++程序设计教程</booktitle></book>
```

2．DOMText 类的方法

DOMText 类派生于 DOMCharacterData 类，这两个类的方法都可以用于 DOMText 对象的文本，如表 12.3 所示。

表 12.3 DOMText 类和 DOMCharacterData 类的方法

方法	说明
isWhitespaceInElementContent()	测试文本节点是否含有空白符。如果元素内容有空白符，则返回 TRUE；否则，返回 FALSE
appendData(string data)	添加字符串数据到文本节点的字符数据的末尾。data 参数为要添加的字符串
deleteData(int offset, int count)	删除文本节点内容中从第 offset 个位置开始的 count 个字符。offset 值从 0 开始编号
insertData(int offset, string data)	在文本节点内容的第 offset 个位置插入一个字符串。data 参数为插入的字符串
replaceData(int offset,int count,string data)	将文本节点内容的第 offset 个位置开始的 count 个字符替换为 data 参数指定的字符串
string substringData(int offset, int count)	返回文本节点内容的第 offset 位置开始的 count 个字符

12.3.6 DOMNodeList 类

DOMNodeList 对象表示有序节点的集合，集合元素的索引号从 0 开始编号，此集合是可变的。它通常是由返回多个节点值的 DOM 方法中返回的值。DOMNodeList 对象的 item() 方法返回 DOMNodeList 对象中以 index 为索引号的节点，其语法如下：

```
item(int index)
```

DOMNodeList 对象的 length 属性存放节点集合的元素个数（即节点数）。

12.3.7 DOMAttr 类

DOMAttr 类派生于 DOMNode 类，DOMAttr 对象用来存放 DOMElement 对象表示的元素的属性。利用其构造方法，可以创建属性节点对象，语法如下：

```
DOMAttr(string name [, string value])
```

其中，name 参数为属性名，value 参数为属性值。

DOMAttr 类的属性如表 12.4 所示

表 12.4 DOMAttr 类的属性

属性名	值类型	是否只读	说明
name	string	是	属性的名字
ownerElement	DOMElement	是	当前属性对象所在的元素
value	string	否	属性的值

【例 12.14】 输出 students.xml 文档中每个学生的个人信息（ex12_14.php）。

```
<?php
$xmlfile = "students.xml";
$dom=new DOMDocument();
$dom->load($xmlfile,LIBXML_NOBLANKS);
$root = $dom->documentElement;      //读取根元素
```

```
    $children = $root->childNodes;      //获取根元素的下一级子节点集
    $count = 1;
    printTree($children);
    echo "XML 文档的元素个数: $count";

    function printTree($nodeCollection) //直接递归调用本方法,输出所有元素及其内容
    {
      global $count;
      echo "<ul>";
      for ($t=0; $t<$nodeCollection->length; $t++)
      {
          $count++;
          $node = $nodeCollection->item($t);
          echo "<li>";
          switch ($node->nodeType) {
              case XML_ELEMENT_NODE:   //元素节点
                  print "元素: ".$node->nodeName;
                  if ($node->nodeName == "NAME") {
                      $id=$node->getAttribute("ID");
                      print " ,ID 属性: ".$id."\n";
                  }
                  break;
              case XML_ATTRIBUTE_NODE:    //属性节点
              print "属性名: ".$node->nodeName;
              print " 属性值: ".$node->nodeValue."\n";
              break;
          case XML_TEXT_NODE:
          case XML_CDATA_SECTION_NODE:
              print "内容: ".$node->nodeValue."\n";
              break;
          default:
              print "其他节点名: ".$node->nodeName."\n";
          }
        $nextCollection = $node->childNodes;
        printTree($nextCollection);     //直接递归调用
      }
      echo "</ul>";
    }
    ?>
```

上述程序生成 students.xml 文件的 DOM 树。利用文档对象的 documentElement 属性获取根节点，然后读取根元素的子节点集（表示每个学生的元素节点），再以子节点集为参数，调用自定义函数 printTree()。而在 printTree()函数中，直接递归调用自身函数，检索各级子节点元素和属性值，把所有元素节点、属性节点、文本节点的名称和内容输出来。此外，还统计 XML 文档的元素个数。输出结果如图 12.5 所示。

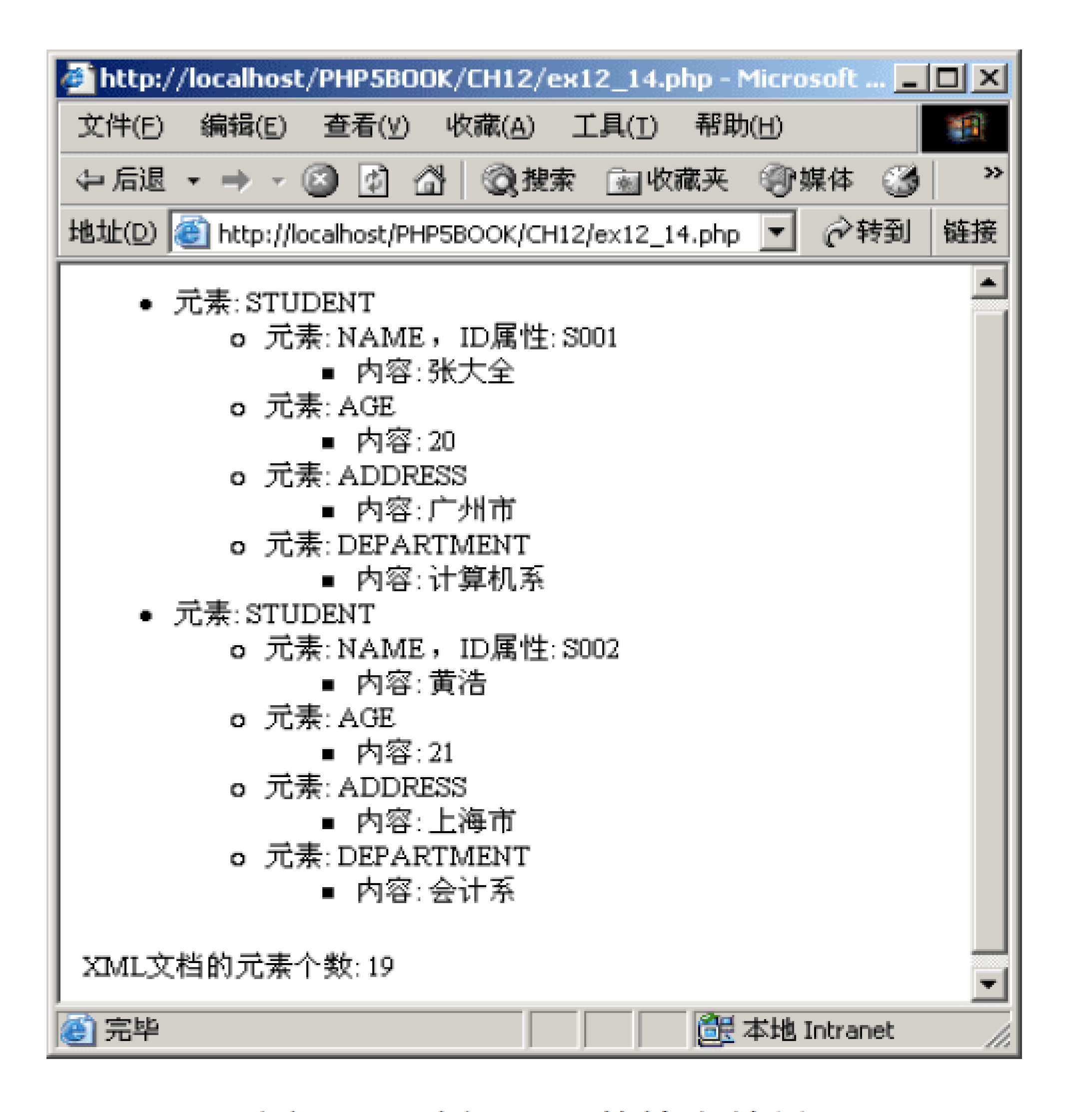


图 12.5　例 12.14 的输出结果

12.3.8　用 DOM 管理 XML 文档

利用 DOM，通过建立含有 XML 文档的 DOM 树来管理 XML 文档。可以修改、遍历 DOM 树，访问 DOM 树的属性、数据、元素。可以通过修改对象的属性和方法来修改 XML 文档。DOM 方法也允许用户查询 XML 文档，从而获取指定的信息。DOM 方法除了解析 XML 文档外，还允许用户建立 XML 文档。

1．修改 XML 文档

可以利用 DOM 方法修改 XML 文档。修改文档包括增加新元素、修改属性值、删除元素等。DOM 方法需要大量内存来解析文档。因此，可以利用 DOM 方法修改较小的 XML 文件，而对较大的 XML 文件，则使用 SAX 方法。

为了理解如何用 DOM 修改 XML 文档，考虑一个关于在线书店购物应用程序的 XML 文档，如例 12.15 所示。

【例 12.15】 建立一个在线购书文件 bookcart.xml。该文档存储的图书信息包括图书编号、书名、作者和单价。

```
<?xml version="1.0" encoding="UTF-8"?>
<Books>
<customer CID="354">
<name>黄洪武</name>
</customer>
<bookInfo>
<book ID="1001">
<bookName>操作系统实用教程</bookName>
<author>李大明</author>
<price>36</price>
```

```
</book>
<book ID="1002">
<bookName>软件工程</bookName>
<author>张玲</author>
<price>25</price>
</book>
</Books>
```

【例 12.16】 在 bookcart.xml 文件中增加一条图书信息（ex12_16.php）。

```
<?php
$xmlfile = "bookcart.xml";
$modfile = AddBook($xmlfile, "1003", "VFP6 程序设计", "徐锋", "27");
print($modfile);

function AddBook($xml, $ID, $bookName, $author, $price)
{
   $dom = new DOMDocument();
   $dom->formatOutput=TRUE;
   $dom->load($xml);
   $root = $dom->documentElement;
   $children = $root->childNodes;
   foreach ($children as $child)
   {
      if ($child->nodeType == XML_ELEMENT_NODE)
      {
         if ($child->nodeName == "bookInfo") {
             $newbook = $dom->createElement("book");
             $newbook->setAttribute("ID", $ID);

             $nname = $dom->createElement("bookName");
             $nametext = $dom->createTextNode($bookName);
             $nname->appendChild($nametext);

             $nprice = $dom->createElement("price");
             $pricetext = $dom->createTextNode($price);
             $nprice->appendChild($pricetext);

             $nauthor = $dom->createElement("author");
             $authortext = $dom->createTextNode($author);
             $nauthor->appendChild($authortext);

             $newbook->appendChild($nname);
             $newbook->appendChild($nauthor);
             $newbook->appendChild($nprice);
```

```
                $child->appendChild($newbook);
            }
        }
    }
    $xml1 = $dom->saveXML();
    $dom->save("addbookcat.xml");
    return $xml1;
}
?>
```

上述代码实现在线书店购物中增加一本新书。调用 AddBook()函数时，需要给该函数传递图书信息参数，如 XML 文件名、图书编号、书名、作者名和单价。AddBook()函数验证文档中元素节点是否为 bookInfo 元素，如果是 bookInfo 元素，则在该元素下增加三个子元素节点和一个属性，表示一本新书信息。用 save()方法将更改后的文档保存到 addbookcat.xml 文件中，并返回 XML 文档的字符串。图 12.6 是例 12.16 的输出结果。

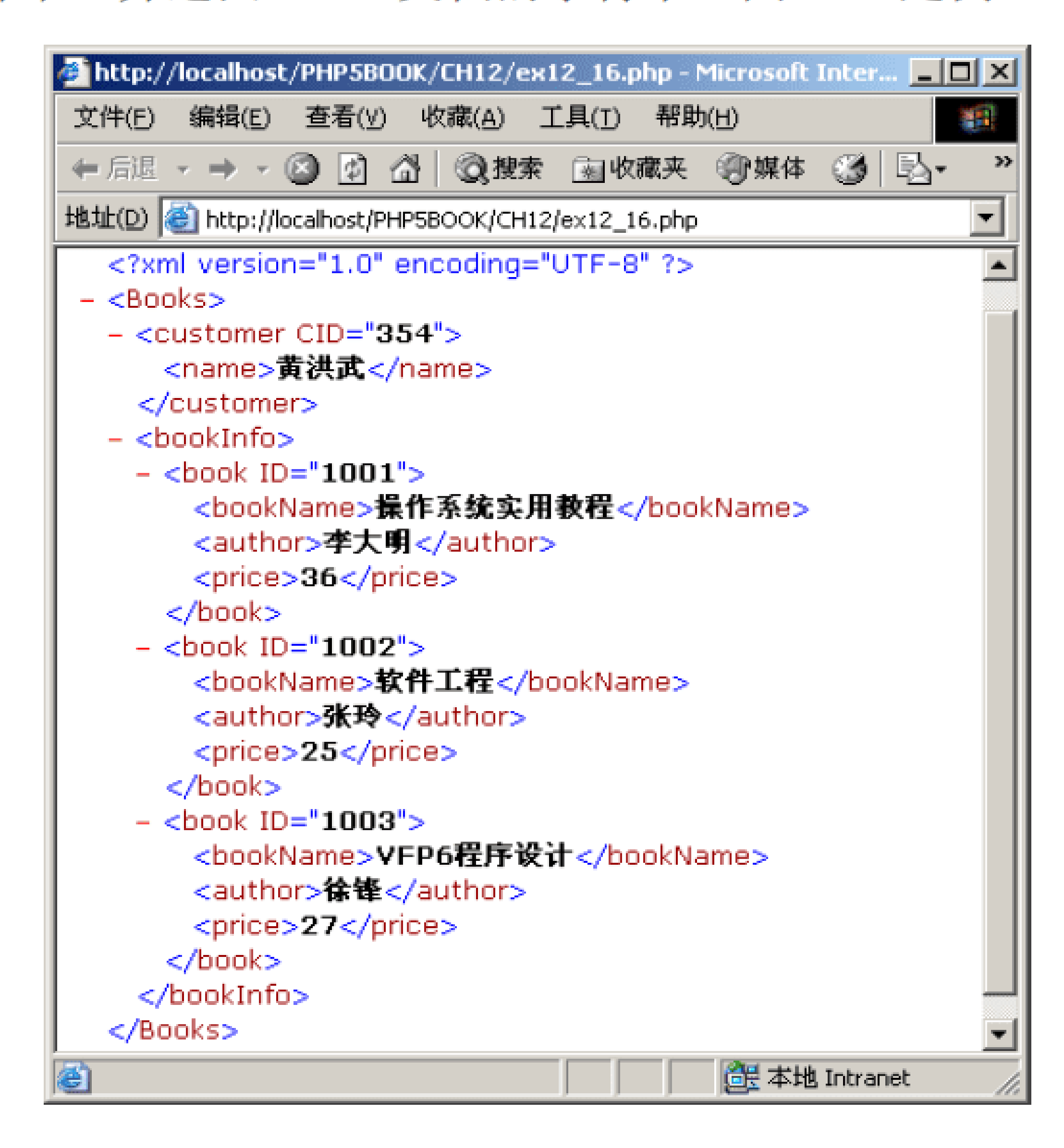


图 12.6　例 12.16 的输出结果

2．用 DOM 删除 XML 文档

可以利用 DOM 的 removeChild()方法，删除 XML 文档的元素节点，比如删除在线书店的图书信息。例 12.17 给出了删除 DOM 树中的元素的方法。

【例 12.17】 删除例 12.16 生成的 addbookcat.xml 文件中图书编号为 1003 的图书信息（ex12_17.php）。

```
<?php
$xmlfile = "addbookcat.xml";
$tt=RemoveBook($xmlfile, "1003");
```

```
print($tt);

function RemoveBook($xml, $id)
{
   $dom = new DOMDocument();
   $dom->load($xml);
   $root = $dom->documentElement; //根元素 books
   $children = $root->childNodes;
   foreach ($children as $child)
   {
      if ($child->nodeType == XML_ELEMENT_NODE)
         {
            if ($child->nodeName == "bookInfo")
            {
               $BookInfo = $child->childNodes;   //book 元素集合
               foreach ($BookInfo as $book)
               {
                  if ($book->nodeType == XML_ELEMENT_NODE)
                  {
                     $ts = $book->getAttribute("ID");
                     if ($ts == $id)
                        $child->removeChild($book);
                  }
               }
            }
         }
    }
    $xml1 = $dom->saveXML();
    $dom->save("remove.xml");
    return $xml1;
}
?>
```

3．替换元素节点内容

利用 DOMNode 类的 replaceChild()方法可以替换元素的子节点，用 DOMText 类的 replaceData()方法替换文本节点的内容。

【例 12.18】 将 XML 文档的 child3 元素替换为 newchild 元素（ex12_18.php）。

```
<?
$doc = DOMDocument::loadXML('<?xml version="1.0"?>
<root>
<child1>child1 content</child1>
<child3>child3 content</child3>
</root>');
$root = $doc->documentElement();
```

```
$child3 = $root->getElementsByTagName("child3")->item(0);
$oldchild = $root->replaceChild(new DOMElement("newchild", "new content"),
$child3);
print $doc->saveXML();
?>
```

在该程序中，创建一个新元素 newchild，并用来替换文档的 child3 元素。该方法返回文档树中被替换的节点（即 child3 元素），输出结果如图 12.7 所示。



图 12.7　例 12.18 的输出结果

4．用 DOM 查询 XML 文档

DOM 除了增加、替换和删除 XML 文档的元素外，还可以检索 XML 文档的指定信息。其检索方法是对 XML 文档，从根元素开始逐层遍历 XML 元素，查找指定元素或者信息。例如，已经建立了一个存储所有学生信息的 XML 文档，学生信息包括姓名、年龄、所在系和家庭地址等。要求从 XML 文档中检索一个指定学生的个人信息，可以建立一个查询，用 DOM 解析器检索 XML 文档中的信息。

【例 12.19】 用 DOM 查询 XML 文档。在 students.xml（见例 12.1）文件中查找姓名为“黄浩”的学生的年龄及所在系（ex12_19.php）。程序如下：

```
<?php
//读取某节点下所有子文本节点的内容
function get($domnode) {
   $content='';
   foreach ($domnode->childNodes as $child) {
      if ($child->nodeType==XML_TEXT_NODE)
         $content.= $child->nodeValue;
   }
   return $content;
}
//主程序
$dom=new DOMDocument(1.0,"UTF-8");
$dom->load("students.xml");
$root=$dom->documentElement;
foreach ($root->childNodes as $element) {
```

```
  if ($element->nodeType==XML_ELEMENT_NODE and $element->nodeName==
  "STUDENT") {
     $c1 = false;
     foreach ($element->childNodes as $child) {
        if ($child->nodeType==XML_ELEMENT_NODE and $child->nodeName==
        "NAME"){
           if (($name=get($child))=="黄浩")
              $c1=true;
        }
        if ($child->nodeType==XML_ELEMENT_NODE and $child->nodeName=="AGE")
           $age=get($child);
        if ($child->nodeType==XML_ELEMENT_NODE and
              child->nodeName=="DEPARTMENT")
           $dept=get($child);
     }
     if ($c1)
        print("姓名: $name <br/> 年龄: $age <br/> 所在系: $dept<br/>");
  }
}
?>
```

访问此程序，输出结果如图 12.8 所示。

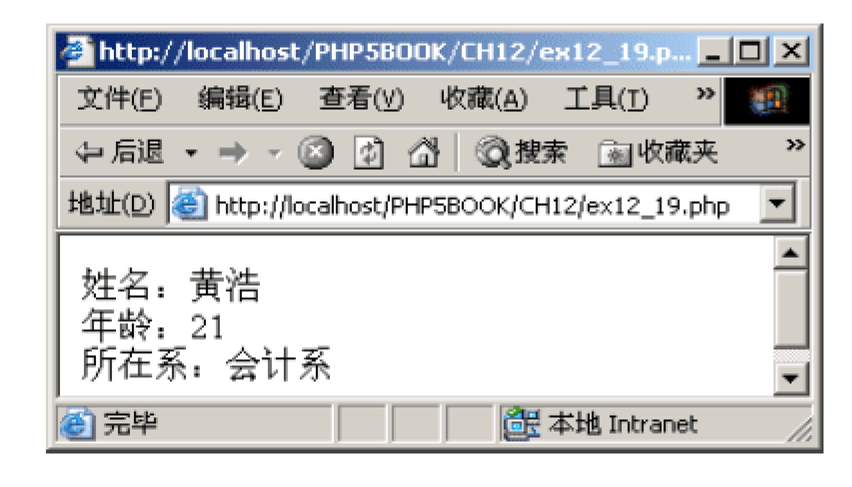


图 12.8　例 12.19 的输出结果

5. 用 DOM 创建 XML 文档

利用 DOM 创建 XML 文档时，首先创建一个 Document 对象，建立一个根元素节点，添加到 Document 对象，然后在根元素节点下创建各级子元素节点，在元素节点下添加属性节点，以构造 DOM 树。

【例 12.20】 用 DOM 创建一个包含图书信息的 XML 文档（ex12_20.php）。

```
<?
$dom = new DOMDocument('1.0','UTF-8');
$dom->formatOutput = TRUE;
$root = $dom->createElement('Books');
$dom->appendChild($root);
//增加第 1 本书
$book=$dom->createElement("book");
$book->setAttribute("ID","1001");
$book->appendchild($dom->createElement("bookName","编译原理"));
$book->appendchild($dom->createElement("author","陈新云"));
$book->appendchild($dom->createElement("price","23.50"));
$root->appendChild($book);
//增加第 2 本书
$book=$dom->createElement("book");
$book->setAttribute("ID","1002");
```

```
$book->appendchild($dom->createElement("bookName","数据结构"));
$book->appendchild($dom->createElement("author","黄英明"));
$book->appendchild($dom->createElement("price","25.00"));
$root->appendChild($book);
//输出 XML 文档
echo $dom->saveXML();
$dom->save("mybooks.xml");
?>
```

访问此程序，输出的 XML 文档如图 12.9 所示。

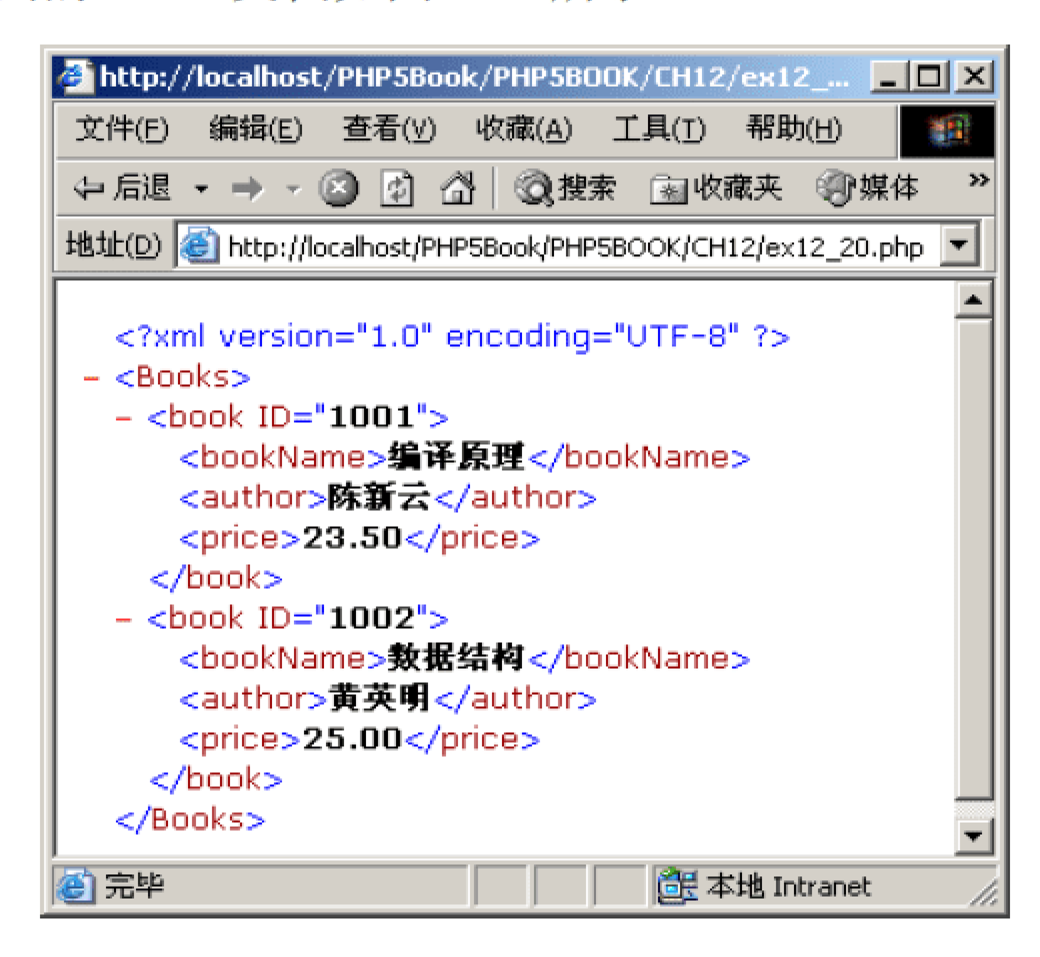


图 12.9　例 12.20 的输出结果

6．验证 XML 文档

第 11 章介绍了验证 XML 文档的三种方法。可以使用这三种方法之一和 DOM 扩展来实现 XML 验证。在装入 XML 文档时，可以使用 LIBXML_DTDVALID 常量和其他装入选项，在解析 XML 文档期间，使用 DTD 来验证 XML 文档的有效性。在使用 DOM 扩展验证文档之前必须先装入 DTD。用带 LIBXML_DTDLOAD 解析选项装入文档和外部 DTD，但解析时不执行验证。通过调用 DOMDocument 对象的 validate()方法来验证 XML 文档的有效性。该方法返回 TRUE 或 FALSE，表示文档的验证状态是否为真。

【例 12.21】 XML 文档的验证（ex12_21.php）。

```
<?
$dom = DOMDocument::loadXML('<?xml version="1.0"?>
<!DOCTYPE courses [
<!ELEMENT courses (course+)>
<!ELEMENT course (title)>
<!ELEMENT title (#PCDATA)>
]>
<courses>
```

```
<course>
<title>算法设计与分析</title>
</course>
</courses>');
$isvalid = $dom->validate();
var_dump($isvalid);
?>
```

运行上述程序后，$isvalid 变量的值为 TRUE，表示该 XML 文档是有效的，符合 DTD 规则。用 var_dump()函数输出$isvalid 变量的类型和值。

12.4 PHP 5 的 SimpleXML 解析器

SimpleXML 扩展是 PHP 5 的另一个基于树的解析器，它提供少量的 API，以一种简单易用的、直观的方式来处理 XML 文档。SimpleXML 扩展只有一个 SimpleXMLElement 类、三个函数和几个类方法。下面以一个 XML 文档为例，介绍如何利用 SimpleXML 扩展来处理 XML 文档。

【例 12.22】 一个简单的 XML 文档（books.xml）。此文档存放的是图书信息，包括书名、作者、简介等信息。

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<library>
    <book id="01001">
        <title>C 语言程序设计教程</title>
        <author gender="女">王英</author>
        <description>这是一本通俗易懂的、例题丰富的教程</description>
    </book>
    <book id="02001">
        <title>编译原理</title>
        <author gender="男">肖意</author>
        <description>这是一本经典的、畅销的编译原理教程</description>
    </book>
</library>
```

12.4.1 创建 SimpleXMLElement 对象

SimpleXMLElement 类是 SimpleXML 扩展的核心类。可以通过以下四种方法来创建该类的对象。

1. 用 new 关键字创建 SimpleXMLElement 对象

SimpleXMLElement 类提供自身的构造方法，其格式如下。

格式：SimpleXMLElement(string data[,int options[, bool data_is_url [, string ns [, bool is_prefix]]]])

该构造方法中，data 参数可以是表示 XML 文档的字符串，也可以是存放 XML 文档的

文件名或 URL。如果 data 参数值是文件名或 URL，则必须将 data_is_url 参数设置为 TRUE。options 参数的值参见 PHP 5 指南的 Libxml 扩展。data_is_url 参数的默认值为 FALSE，将其设为 TRUE，要求 data 参数为文件名或 URL。

用 new 关键字初始化 SimpleXMLElement 类的构造方法，可以创建 SimpleXMLElement 类的对象实例。例如，以下程序以 XML 字符串为参数，创建 SimpleXMLElement 对象：

```
<?
$xml = "<root><node1>content</node1></root>";
$sxe = new SimpleXMLElement($xml);
var_dump($sxe);
?>
```

程序中，$sxe 变量的类型为 SimpleXMLElement 对象。用 var_dump()函数输出$sxe 变量的类型和值，输出结果如下：

```
object(SimpleXMLElement)#1 (1) { ["node1"]=> string(7) "content" }
```

又如，利用 books.xml 文件作为参数，创建 SimpleXMLElement 对象，程序如下：

```
<?
$sxe = new SimpleXMLElement("books.xml",NULL,TRUE);
echo $sxe->asXML();
?>
```

该程序的输出结果与例 12.22 的 books.xml 文件内容相同。

2．simplexml_load_file()

函数格式：object simplexml_load_file(string filename)

该函数装入由 filename 参数指定的 XML 文件，返回一个 SimpleXMLElement 对象。如果装入文件出错，则返回 FALSE。例如：

```
<?php
$xml = simplexml_load_file("books.xml");
var_dump($xml);
?>
```

返回值如下：

```
object(SimpleXMLElement)#1 (1) {
  ["book"]=>
  array(2) {
    [0]=>
    object(SimpleXMLElement)#2 (4) {
      ["@attributes"]=>
      array(1) {
        ["id"]=>
        string(5) "01001"
      }
```

```
        ["title"]=>
        string(25) "C语言程序设计教程"
        ["author"]=>
        string(6) "王英"
        ["description"]=>
        string(51) "这是一本通俗易懂的、例题丰富的教程"
      }
      ⋮
    }
  }
```

从返回值可以看出，simplexml_load_file()函数的返回值是 SimpleXMLElement 对象。对于文档中的多个相同名字的元素，以数组形式表示。比如<book>元素以含有两个数组元素的数组表示。注意，显示的 XML 并不包括 author 元素的属性。为了显示其属性，需要用 attributes()方法。

3．simplexml_load_string()函数

函数格式：object simplexml_load_string (string data)

该函数装入由字符串表示的 XML 文档，返回一个 SimpleXMLElement 对象。data 参数是一个存放 XML 文档的字符串。

该函数与 simplexml_load_file()在用途上是一样的，除了输入参数是字符串而不是文件名以外。例如：

```
<?
$xml = "<root><node1>content</node1></root>";
$sxe = simplexml_load_string($xml);
var_dump($sxe);
?>
```

返回值为：

```
object(SimpleXMLElement)#1 (1) {
  ["node1"]=>
  string(7) "content"
}
```

4．simplexml_import_dom()

函数格式：SimpleXMLElement simplexml_import_dom(DOMNode node)

该函数将 DOM 文档的一个节点转换为 SimpleXMLElement 对象。node 参数是 DOM 扩展的 DOMNode 类型的对象。例如：

```
<?php
$dom = new domDocument();
$dom->loadXML('<books><book><title>PHP and XML</title></book></books>');
if (!$dom) {
    echo '解析文档遇到错误';
```

```
        exit;
    }
    $sxe = simplexml_import_dom($dom);
    echo $sxe->book[0]->title; //输出 PHP and XML
    ?>
```

程序中，simplexml_import_dom()函数的值为一个 SimpleXMLElement 对象，然后输出该对象的 title 元素值。

根据 XML 文档，创建了 SimpleXMLElement 对象后，为了访问 XML 文档的元素，只需将文档的元素看作是 SimpleXMLElement 对象的属性，按照属性的引用方法，即可访问 XML 文档中相应的元素和内容。如果文档的同一层次中有多个元素名是相同的，那么将这些元素作为对象的属性数组，其数组元素下标从 0 开始编号。

【例 12.23】 输出 books.xml 文件中第 1 个 book 元素的书名、作者和简介。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>第一本书</title>
</head>
<body>
<?
$sxe = simplexml_load_file("books.xml");
$title=$sxe->book[0]->title;   //读取第 1 本书的信息
$author=$sxe->book[0]->author;
$description=$sxe->book[0]->description;
echo "书名：".$title."<br>";
echo "作者：".$author."<br>";
echo "简介：".$description."<br>";
?>
</body>
</html>
```

由于 books.xml 文件中有两个 book 元素，因此，程序中的 book[0]表示第 1 个 book 元素，$sxe->book[0]->title 返回第 1 个 book 元素的 title 元素内容，其余类似理解。程序输出结果如图 12.10 所示。

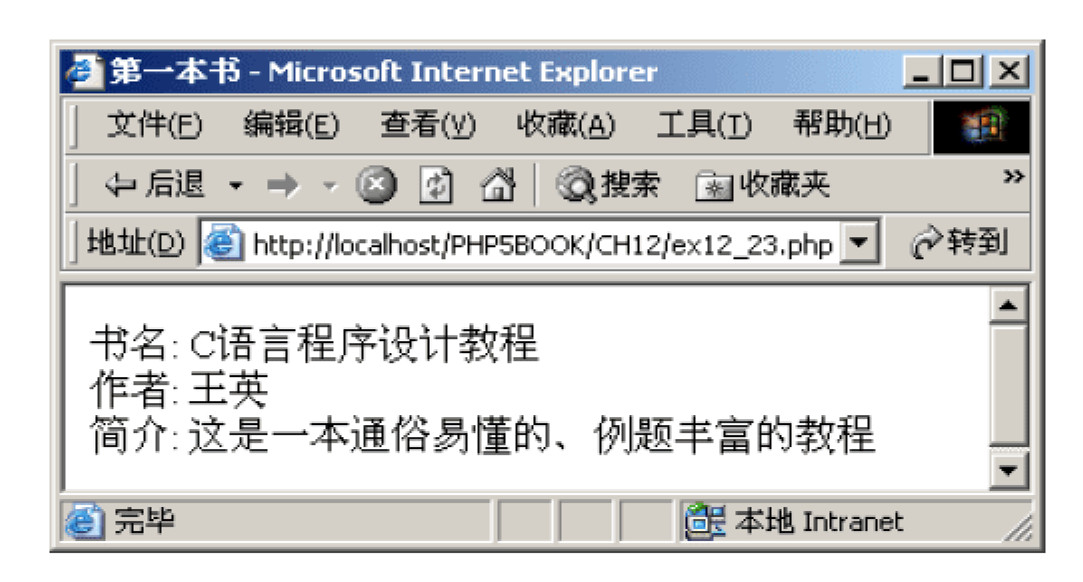


图 12.10　例 12.23 的运行结果

12.4.2　SimpleXML 方法

一旦利用上述四个函数之一把 XML 文档装入到一个 SimpleXMLElement 对象中，就可以利用 SimpleXMLElement 类的方法操作 XML 文档的元素和属性。SimpleXMLElement 类的常用方法如下。

1．AddAttribute()

格式：addAttribute(string name, string value[, string namespace])

该方法给 SimpleXMLElement 对象增加一个属性。name 参数指定要增加的属性名，value 参数指定该属性的值，可选的 namespace 参数指定该属性所属的名字空间。

2．AddChild()

格式：SimpleXMLElement addChild(string name [, string value [, string namespace]])

该方法增加一个子元素到 XML 元素节点，其返回值为子元素的 SimpleXMLElement 对象。name 参数指定要增加的元素名，value 参数指定新增元素的值，namespace 参数指定新增元素所属的名字空间。

【例 12.24】 给 books.xml 增加一本新书（ex12_24.php）。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>增加一本新书</title>
</head>
<?php
$sxe = new SimpleXMLElement("books.xml",NULL,TRUE);
$book=$sxe->addChild("book");
$book->addAttribute("id","02002");
$book->addChild("title", "PHP 5 Web开发技术");
$author=$book->addChild("author","徐辉");
$book->addChild("description","这是一本介绍 PHP 5 和 XML 结合的新书");
$author->addAttribute('gender', '男');
echo $sxe->asXML();
$sxe->asXML("newbooks.xml");
?>
</html>
```

访问该程序，输出的结果包括了 books.xml 文件的内容以及新增加的一本书的信息。并将这些内容保存到 newbooks.xml 文件。

3．attributes()

格式：SimpleXMLElement attributes()

该方法返回当前 XML 元素节点的属性及其属性值。其返回值是 SimpleXMLElement 对象。

XML 元素的属性提供关于 XML 元素的一些附加信息。在例 12.22 所示的 books.xml 文件中，book 元素含有一个 id 属性，表示图书编号，author 元素含有 gender 属性，表示

作者的性别。可以使用 attributes()方法，检索这些元素的属性。

【例 12.25】 输出 books.xml 文件中每本书的作者的性别（ex12_25.php）。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>显示作者信息</title>
</head>
<body>
<?php
$sxe = simplexml_load_file("books.xml");
foreach($sxe->book as $book) {
  echo $book->author." 是 ".$book->author->attributes()."<br />";
}
?>
</body>
</html>
```

输出结果如下：

```
王英 是 女
肖意 是 男
```

也可以直接引用特定元素的属性。例如，books.xml 文件中有两个<book>元素，为了查看第 2 个<book>元素的作者的性别，代码为：

```
echo $sxe->book[1]->author->attributes();
```

输出结果为：男

通常一个 XML 元素节点含有多个属性，例如，假设第一本书的 author 元素如下：

```
<author gender="女" age="35">王英</author>
```

那么，$sxe->book[0]->author->attributes()返回的结果是一个含有两个元素的数组。可以用 for 循环语句输出该元素的所有属性，如下：

```
foreach($sxe->book[0]->author->attributes() AS $a => $b) {
echo "$a = $b <br />";
}
```

输出结果为：

```
gender = 女
age = 35
```

4．asXML()

格式：mixed asXML([string filename])

该方法返回基于 SimpleXML 对象的格式良好的 XML 字符串。返回值是整个 XML 文档或者其子树的字符串。如果不给出 filename 参数，则返回该 XML 字符串。如果给出

filename 参数，则将文档或子树内容保存到指定文件中。

【例 12.26】 输出 books.xml 文件的文档内容，并将其第一个 book 元素及其子元素保存到 book1.xml 文件（ex12_26.php）。

```
<?php
$sxe = simplexml_load_file("books.xml");
echo htmlspecialchars($sxe->asXML());
$sxe->book[0]->asXML("book1.xml");
?>
```

5. children()

格式：SimpleXMLElement children()

该方法返回当前元素节点的子节点，返回值是一个 SimpleXMLElement 对象。

【例 12.27】 用 children()方法输出 books.xml 的第一本书的子元素内容。

```
<?php
$xml = simplexml_load_file("books.xml");
foreach($xml->book[0]->children() AS $child) {
echo "$child <br />";
}
?>
```

程序的输出结果为：

```
C 语言程序设计教程
王英
这是一本通俗易懂的、例题丰富的教程
```

6. xpath()

格式：array xpath(string path)

xpath 是一个 W3C 标准，提供一个直观的、基于路径的标识 XML 节点的语法。例如，针对 books.xml 文档，可以用路径表达式/library/book/author 检索所有 author 节点。xpath 也提供一组函数，用于检索满足条件的节点。

xpath()方法在 XML 文档中检索与 path 参数指定的 xpath 路径表达式相匹配的节点。其返回值是 SimpleXMLElement 对象数组。

【例 12.28】 检索 books.xml 文件中的所有图书的作者（ex12_28.php）。

```
<?php
$xml = simplexml_load_file("books.xml");
$authors = $xml->xpath("/library/book/author");
foreach($authors AS $author) {
  echo $author."<br>";
}
?>
```

输出结果如下：

```
王英
肖意
```

也可以用 xpath()函数根据指定的值，有选择地检索一个节点以及它的子节点。例如，要检索作者为“王英”的所有图书的书名。程序如下：

```
<?php
$sxe = simplexml_load_file("books.xml");
$book = $sxe->xpath("/library/book[author='王英']");
echo $book[0]->title;
?>
```

输出结果为：

```
C 语言程序设计教程
```

12.4.3 SimpleXML 的应用

利用 SimpleXMLElement 对象，可以很方便地对 XML 文档进行读取元素内容、增加元素、修改元素内容、删除元素等操作。前面的例子已经介绍了如何读取元素、增加元素。为了修改元素内容，可以通过给 SimpleXMLElement 对象的相应属性赋值来实现。

【例 12.29】 修改 books.xml 文件中第 2 个 book 元素节点的 description 子元素的内容（ex12_29.php）。

```
<?
$sxe = simplexml_load_file("books.xml");
$sxe->book[1]->description="本书介绍编译原理的一般理论和常用方法";
echo $sxe->asXML();
?>
```

也可以用 SimpleXMLElement 对象，删除 XML 文档的元素节点。为此，使用 PHP 内置的 unset()函数。unset()函数的参数必须是已装载的 SimpleXMLElement 对象的某一个属性，即由属性指定要删除的元素。

【例 12.30】 删除 books.xml 文件中第 2 个 book 元素节点的 description 子元素。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>用 SimpleXML 删除元素节点</title>
</head>
<body>
<?
$sxe = simplexml_load_file('books.xml');
unset($sxe->book[1]->description);    //删除 description 元素
print $sxe->asXML();
?>
```

```
</body>
</html>
```

除了利用 SimpleXMLElement 对象来管理 XML 文档的元素以外，还可以对 XML 文档中元素的属性进行读取属性值、修改属性值、增加属性和删除属性等操作。当访问属性时，可以用数字（从 0 开始编号）或属性名作为数组的下标。

【例 12.31】 输出 books.xml 文件中第 1 个 book 元素的 id 属性值及其作者的性别。程序如下：

```
<?
$sxe = simplexml_load_file('books.xml');
echo $sxe->book[0]["id"].",";
echo $sxe->book[0]->author["gender"];
?>
```

程序中，book[0]["id"]表示第 1 个 book 元素的 id 属性，author["gender"]表示 author 元素的 gender 属性。该程序输出的结果是：01001，女。

修改属性内容的方法与修改元素内容的方法相同。只需将 SimpleXMLElement 对象的属性设置为一个字符串，从而改变了元素属性的值。

【例 12.32】 将 books.xml 的第 2 个 book 元素的 id 属性值改为“04001”。

```
<?
$sxe = simplexml_load_file('books.xml');
$sxe->book[1]['id'] = "04001";
print $sxe->book[1]['id'];
echo $sxe->asXML();
?>
```

可以用 unset()函数删除 XML 文档中元素的属性，它与删除 XML 文档的元素的方法一样。例如，删除 books.xml 的第 2 个 book 元素的 id 属性，程序如下：

```
<?
$sxe = simplexml_load_file('books.xml');
unset($sxe->book[1]['id']);
echo $sxe->asXML();
?>
```

实　验　12

1．利用 DOM 扩展的编程技术，编写一个程序实现：创建一个存放商品销售清单的 XML 文档，它包含销售单编号、商品名、销售日期、单价、销售量等信息，文件名为 sale.xml，其内容如下：

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<sales>
  <sale ID="1001">
```

```
      <productName>电视机</productName>
      <date>2006-10-20</date>
      <price>3200</price>
      <quantity>5<quantity>
    </sale>
    <sale ID="1002">
      <productName>冰箱</productName>
      <date>2006-10-20</date>
      <price>2400</price>
      <quantity>3<quantity>
    </sale>
  </sales>
```

2．利用 PHP 5 的 SAX 解析器，编程将 sale.xml 文件的内容转换为 HTML 表格形式输出。

3．分别用 PHP 5 的 DOM 和 SimpleXML 解析器编程技术，编写以下四个问题的程序，实现对 sale.xml 文件的操作。

（1）增加一个新的销售清单。其销售单编号为 1004，商品名为“洗衣机”、销售日期为 2006 年 10 月 23 日，单价为 2100 元、销售量为 7 台。

（2）删除销售单编号（即 ID 属性）为 1002 的<sale>元素。

（3）在 ID 属性为 1002 的<sale>元素中，将销售量< quantity >元素的值改为 6。

（4）查询 ID 属性为 1002 的销售清单信息，输出其商品名、单价和销售量。

习　题　12

1．PHP 5 处理 XML 文档的技术有哪些？

2．什么是 XML 解析器？XML 解析器分为几种类型？

3．什么是 SAX 解析器？如何理解 SAX 解析器的工作原理？

4．如何理解 DOM 解析器的工作原理？

5．如何理解 SimpleXML 解析器的工作原理？

6．比较 DOM 解析器和 SAX 解析器的差异。

7．比较 DOM 解析器和 SimpleXML 解析器的差异。

第 13 章　基于 PHP 的 Web 服务

本章导读

Web 服务是一项当前最流行的应用程序开发技术。Web 服务实质上是一个提供某种功能的程序代码，它允许在本地网络和 Internet 上使用“远程过程调用（RPC）”机制，使得一个 Web 服务器的应用程序（称为客户端程序）通过 HTTP 和 XML 等平台无关标准，对运行于远程主机 Web 服务器的程序（称为 Web 服务）进行调用，客户端程序只需要知道 Web 服务的 URL 和方法调用使用的数据类型，而不需要知道运行 Web 服务的操作系统和所采用的编程语言。因此，Web 服务从根本上说是一种 Web 环境下跨平台、跨语言、松耦合的分布式系统最佳解决方案，可以实现不同 Web 服务器的应用程序之间的交互功能。这种交互需要使用 XML-RPC 或者 SOAP（简单对象访问协议）协议的支持，对数据进行打包、发送和接收。

XML-RPC 和 SOAP 是基于 XML 和 HTTP 实现远程过程调用的两种主要方法。XML-RPC 的目标是寻求以简单而有效的方式来请求和接收信息，SOAP 的主要设计目标也在于简单性和可扩展性。SOAP 弥补了 XML-RPC 的一些不足，有重要的安全性机制和鲁棒的对象模型。而 XML-RPC 则具有稳定的代码、更简单的接口以及更少的开销，可以使用更少的处理资源来做同样工作，实现更为容易。

本章将分别介绍 XML-RPC 和 SOAP 协议的工作原理，以及如何利用这两种协议和 PHP 5 结合起来，构建 Web 服务和 Web 服务的客户端程序的编程技术。

13.1　XML-RPC 工作原理

13.1.1　RPC 概述

RPC（远程过程调用）是一个标准编程接口，它采用客户机-服务器模式，调用程序（称为客户机）通过网络，将消息传输给服务器的方法或者过程，执行过程并返回结果给调用程序，客户端调用程序接收响应信息，获得结果，然后调用程序执行继续进行。RPC 体系结构如图 13.1 所示。调用远程过程的客户端可能与服务器位于同一主机上，也可能位于网络中不同的主机上。

XML-RPC 和 SOAP 是两个最流行的 RPC 协议，它们都是通过 Web，在 HTTP 协议层之上传递 RPC 消息的协议，两者都能通过防火墙传递消息。但两者是有差别的，XML-RPC 的初始化工作比 SOAP 更简单；XML-RPC 产生的请求和响应信息比 SOAP 更小，解析时间更短；SOAP 允许用 Schema 定义用户自定义类型，这允许 XML 和 PHP 之间进行数据

验证，自动地转换数据类型。在 XML-RPC 中，必须手工执行所有的数据序列化。SOAP 得到 IBM、微软等许多大公司的支持；SOAP 是通用的、可扩充的工具，而 XML-RPC 是有着相对严格定义的特殊协议。

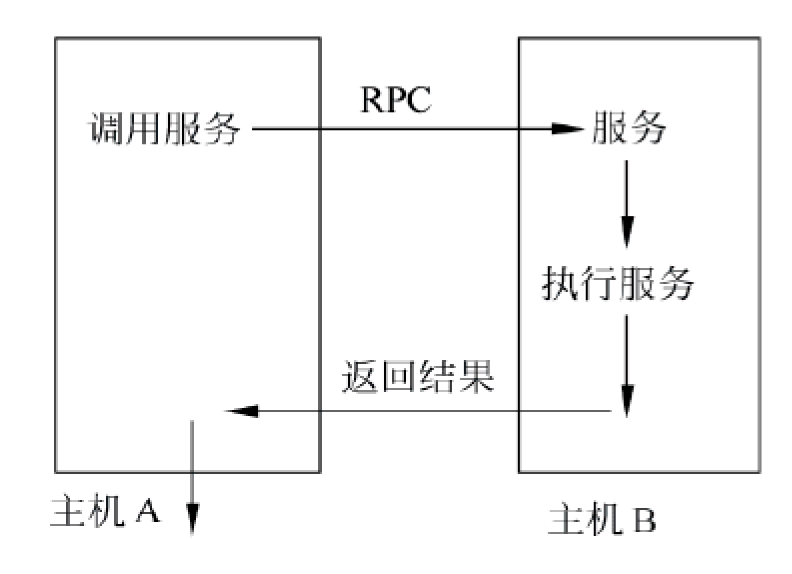


图 13.1　RPC 网络通信体系结构

13.1.2　XML-RPC 协议的工作原理

XML-RPC 即 XML 远程过程调用，是一种规范和一组实现方案，允许运行于不同操作系统、不同平台环境下的软件通过 Internet 进行远程过程调用。例如，XML-RPC 允许 perl 函数调用 Python 的方法，Java Servlet 调用 PHP 的函数。XML-RPC 使用 HTTP 作为传输协议，XML 为传送信息的编码格式。客户端将 XML-RPC 请求编码为 XML 格式，发出 HTTP POST 请求，将请求消息传输给服务器。HTTP POST 请求告诉服务器，客户端准备好传输数据。服务器调用请求的方法，并传递参数给该方法。服务器上的方法返回一个响应，服务器将响应编码为 XML 格式，并将 XML 格式的响应返回给客户端。客户端解析 XML 包，获取返回的值，远程方法返回的值称为服务器响应。XML-RPC 的体系结构如图 13.2 所示。

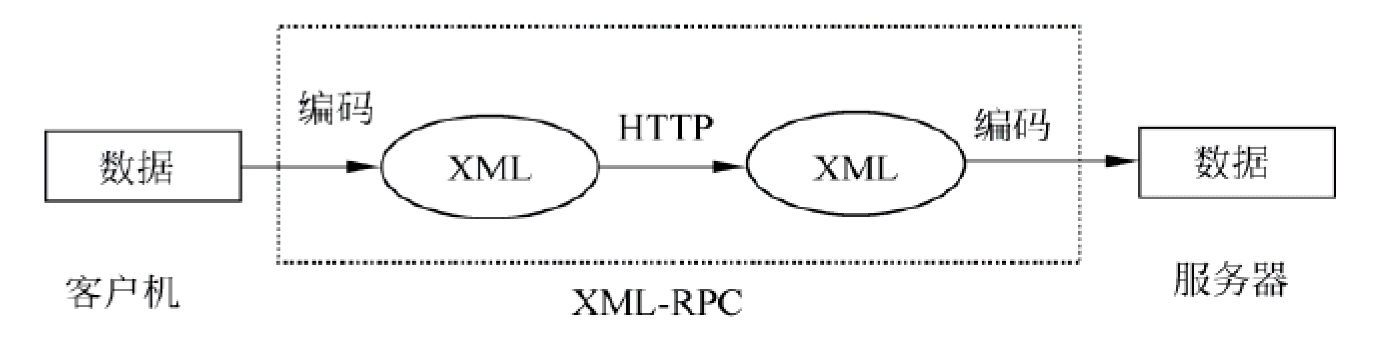


图 13.2　XML-RPC 体系结构

由于 XM-RPC 使用 HTTP 协议，因此 XML-RPC 客户端的请求是同步的、无状态的。XML-RPC 客户端请求的同步是指服务器立即响应客户端的请求，XML-RPC 服务器与客户端在 HTTP 层上传输响应。XML-RPC 客户端请求的无状态是指客户端从一个请求到另一个请求之间不保存任何信息。例如，如果一个客户端调用服务器的远程方法，接收服务器的响应，再调用该服务器的同一个方法，那么客户端的两次请求是不同的请求。

XML-RPC 客户端在接收到服务器的响应之前，不执行任何其他函数。一个客户请求只能调用一个远程方法。

XML-RPC 请求和响应都由 HTTP 头部和 XML 文档体组成。XML 文档体包含着在远程服务器上执行的方法和这些方法所使用的全部参数，返回的响应也是 XML 文档。

一个远程过程的参数和返回值可以是简单的数据形式，例如，逻辑型、整型、字符串、浮点型和日期型，也可以是较复杂的数据结构，例如，通过数组和结构表示的复杂的记录和列表结构。事实上，二进制数同样能被 Base64 编码表示，例如，可以调用远程方法，返回一个图像。

XML-RPC 规范包含了 HTTP 报头、XML-RPC 请求、有效负载格式、XML-RPC 响应和错误处理。

1．HTTP 报头

HTTP 报头指定远程服务器中处理 XML-RPC 请求的程序所在的 URI（统一资源标识

符）、主机名和 XML-RPC 请求的长度等信息。HTTP 报头的格式如下：

```
POST /xmlrpc/xmlrpc_server.php HTTP/1.0
User-Agent: xmlrpc-epi-php/1.0 (PHP)
Host: localhost:80
Content-Type: text/xml
Content-length: 191
```

其中，第一行指定 XML-RPC 请求必须通过 HTTP 协议的 POST 方法来传递。必须指定 User-Agent（用户代理）和 Host（主机）的值。Content-length 给出了 XML-RPC 请求内容的长度（字节）。

2．XML-RPC 请求

在 HTTP 报头之后是 XML-RPC 请求。XML-RPC 请求是一个 XML 文档，表示要发送的 XML-RPC 请求数据。XML-RPC 请求的样式如下：

```
<?xml version='1.0' encoding="UTF-8" ?>
<methodCall>
<methodName>greet</methodName>
<params>
  <param>
    <value><string> Wang Hui</string></value>
  </param>
</params>
</methodCall>
```

全部的请求包含在一个<methodCall>元素中，它包含了请求数据，<methodCall>元素必须包含一个子元素<methodName>，该子元素包含了一个字符串，它描述了一个将被激活的远程方法的名字。如果该方法含有参数，则<methodCall>元素必须包含一个<params>元素，该元素包含若干个<param>子元素，每一个<param>元素又包含<value>元素，在其中给出参数的类型和值。本例中，“Wang Hui”为字符串，因此，它用<string>元素括住。

XML-RPC 的参数的值可以是标量、数组和结构，而标量可以是整型、逻辑型、字符串型、双精度型、日期时间型和 Base64 编码型。XML-RPC 的数据类型、相应元素名及其含义如表 13.1 所示。

表 13.1　XML-RPC 数据类型

XML-RPC 类型	类型说明	对应 PHP 类型	例子
<i4> 或 <int>	4 字节有符号整数	Integer	10
<boolean>	布尔型，值为 0（假）或 1(真)	Boolean	1
<string>	字符串型，默认字符编码为 UTF-8	String	Hello, DevShed!
<double>	带符号的双精度浮点数	Double 或 float	–21.2544
<dateTime.iso8601>	日期时间，格式为 YYMMDDTHH:MM:SS		20011219T12:05:26
<base64>	Base64 编码的二进制数，用来传送二进制信息，如图像	Base64 编码的字符串	eW91IGNhbid0IHJlYWQgdGhpcyE=
<struct>	结构	关联数组	array('color'=>'red','number'=2)
array	数组	数组	array(1,2,'red')

XML-RPC 的数组用<array>元素来描述。一个<array>元素包含一个<data>元素，<data>元素又包含任意多个内含参数的<value>元素，表示各个数组元素的值。一个数组可以包含不同类型的元素值。数组适用于两种情况：一是将要传送的数据由一系列参数所组成，而这些参数在设计时并不知道；二是参数很多，希望以数组形式传送而不是采用分散的个体形式。例如：

```
<array>
  <data>
    <value><boolean>0</boolean></value>
    <value><int>9</int></value>
    <value><string>Hello</string></value>
  </data>
</array>
```

XML-RPC 的结构类型用<struct>元素来描述。每个<struct>元素包含一个或多个<member>元素，表示结构的成员，每个<member>包含一个<name>和一个<value>子元素，分别表示成员的名字和值。例如，下面是一个由两个成员构成的结构。

```
<struct>
  <member>
    <name>name</name>
    <value><string>Wang Hui</string></value>
  </member>
  <member>
    <name>age</name>
    <value><int>38</int></value>
  </member>
</struct>
```

3．XML-RPC 响应

XML-RPC 服务器执行调用方法后，返回给客户端的是 XML-RPC 响应，其编码格式为 XML，它是一个 HTTP 响应，内容类型为 text/xml，XML-RPC 的响应格式如下：

```
HTTP/1.1 200 OK
Connection: close
Content-Length: 191
Content-Type: text/xml
Date: Sat, 07 May 2005 08:12:58 GMT
Server: Apache/2.0.52 (Win32)
<?xml version='1.0' encoding="UTF-8" ?>
<methodResponse>
<params>
  <param>
    <value><string>Hello,Wang Hui</string></value>
  </param>
```

```
</params>
</methodResponse>
```

XML-RPC 响应分为两部分：第一部分是正常的 HTTP 响应报头，首行是状态码为“200 OK”的状态信息，Content-Length 给出了响应正文的长度，Content-Type 是 text/xml，Date 和 Server 是任意的；第二部分是响应正文。响应正文是一个 XML 文档，它包含一个<methodResponse>元素，<methodResponse>元素包含一个唯一的<params>元素，<params>元素又包含唯一的<value>元素，由<value>元素给出返回值。

4．出错响应

如果服务器执行方法时出现错误，则返回一个错误响应信息。在<methodResponse>元素里包含一个<fault>元素，该<fault>元素又包含一个<value>元素，<value>元素包含一个含有两个成员数据的<struct>元素，一个用<int>元素表示的错误代码（faultCode），另一个是字符串型的错误信息（faultstring）。出错信息的 XML-RPC 响应正文的格式如下：

```
<?xml version='1.0' encoding="UTF-8" ?>
<methodResponse>
<fault>
  <value>
   <struct>
    <member>
      <name>faultCode</name>
      <value> <int>4</int> </value>
    </member>
    <member>
     <name>faultString</name>
     <value> <string>Too many parameters.</string> </value>
    </member>
   </struct>
  </value>
</fault>
</methodResponse>
```

13.1.3 使用 XML-RPC 的 Web 服务

Web 服务是一个运行于 Web 服务器上的应用程序，Web 客户也是一个运行在 Web 服务器的应用程序。两者可以位于同一主机，也可以位于网络中不同的主机。Web 客户使用 HTTP 传输层调用远程过程。Web 服务的数据交换过程与 Web 浏览器的相似。Web 浏览器以 HTML 表单的形式传输一个请求给 Web 站点，获取响应的 Web 页面。Web 服务使用 XML 格式而浏览器采用 HTML 格式来传输数据。

XML-RPC 协议应用于 Web 服务解决方案时，它在 Web 客户和 Web 服务之间扮演中间件的角色。XML-RPC 接受 Web 客户端请求，把请求编码为 XML 格式，通过 HTTP 协议传输给远端的 Web 服务。使用 XML-RPC 协议访问 Web 服务的过程如图 13.3 所示。

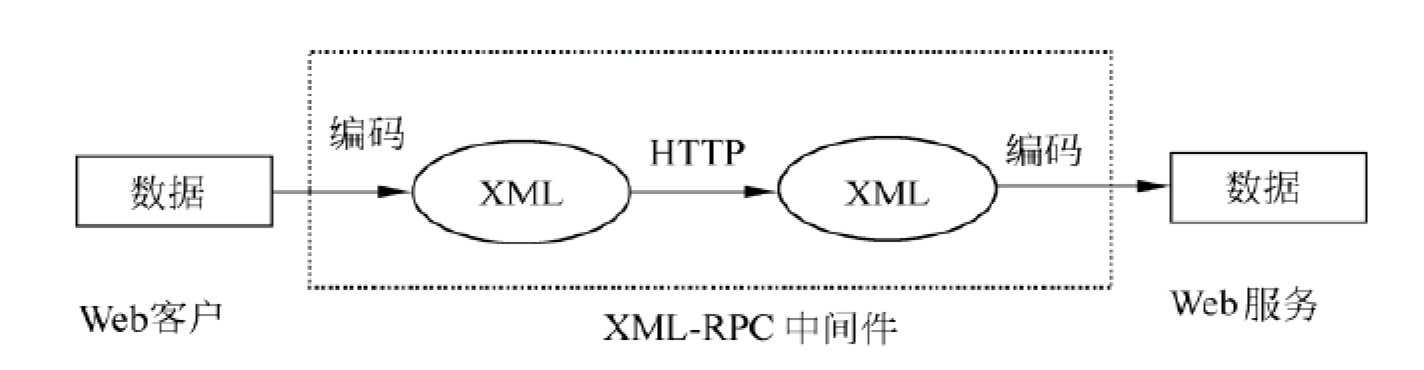


图 13.3 利用 XML-RPC 协议访问 Web 服务

13.2 基于 PHP 5 和 XML-RPC 的 Web 服务

13.2.1 支持 XML-RPC 的 PHP 5 配置

XML-RPC 是 PHP 5 的扩展模块。默认情况下，在 PHP 5 中不能使用 XML-RPC 扩展。在 Linux 和 UNIX 系统下，需要使用“--with-xmlrpc= XMLRPC 目录”配置选项来编译 PHP，才能使用 XML-RPC 扩展功能。

在 Windows 系统中，PHP 5 的 XML-RPC 以动态链接库文件的形式存放在 PHP 5 安装目录的 ext 子目录中。只需修改 PHP 配置文件 php.ini 的 extension_dir 参数值，将其值设置为扩展动态库的安装路径（比如 C:/php5/ext/），并增加一行参数 extension，如下所示：

```
extension_dir = "C:/php5/ext/"
extension = php_xmlrpc.dll
```

13.2.2 编写基于 PHP 5 和 XML-RPC 的 Web 服务程序

利用 PHP 5 实现基于 XML-RPC 的 Web 服务，实质上是用 PHP 编写一些用户自定义函数，将这些自定义函数注册为 Web 服务的方法，供 Web 客户端程序调用。PHP 5 的 XML-RPC 扩展提供如下编写 Web 服务的函数。

1．xmlrpc_server_create()函数

格式：handle xmlrpc_server_create()

该函数创建一个 XML-RPC 服务器，其返回值是：成功时则返回新创建的服务器句柄，失败时返回 false。

2．xmlrpc_server_destroy()函数

格式：void xmlrpc_server_destroy(handle server)

该函数销毁服务器资源。如果不调用此函数，则服务器将在请求结束时销毁资源。server 参数是由 xmlrpc_server_create()函数创建的服务器句柄。

3．xmlrpc_server_register_method()函数

格式：int xmlrpc_server_register_method(handle server, string method_name, string function)

该函数将一个 PHP 自定义函数注册为 Web 服务的方法。成功时返回值为 true，失败时为 false。

参数说明：server 是由 xmlrpc_server_create()函数创建的服务器句柄，method_name 是

要定义的 Web 服务的方法名，function 是代表 Web 服务方法的 PHP 自定义函数。

4．xmlrpc_server_call_method()函数

格式：array xmlrpc_server_call_method(handle server, string xml, mixed user_data [，array output_options])

该函数接收 XML-RPC 请求，调用由 XML-RPC 请求指定的 Web 服务方法。返回值是方法调用的结果，它可以是一个 PHP 类型的值，或者 XML 编码，这依赖于 output_options 参数的设置。

参数说明：server 是创建的服务器句柄，xml 是原始的 XML 请求字符串，user_data 是传递给方法处理函数的任何数据。output_options 是可选的，用于指定 XML 生成选项，其值如下所示。

- output_type：返回的数据为 PHP 类型或者 XML 编码。如果其值为 php，则其他值被忽略，默认为 xml。
- verbosity：检测生成的 XML 文档的紧凑性，其值为 no_white_space、newlines_only 或 pretty 之一，默认值为 pretty。
- version：给出所用的 XML 词汇表的版本。它支持 xmlrpc、soap 1.1 和 simple。关键字 auto 自动检测版本号，默认值为 auto，即 xmlrpc。
- encoding：指定数据的编码。由于 PHP 默认的编码是 iso-8859-1，因此，通常要利用此选项来改变编码，默认为 iso-8859-1。

例如，以下语句定义 output_options 参数的值。

```
$output_options = array(
    "output_type" => "xml",
    "verbosity" => "pretty",
    "version" => "xmlrpc",
    "encoding" => "utf-8"
);
```

利用 PHP 编写基于 XML-RPC 的 Web 服务程序，其编程步骤如下：

（1）定义一些作为 Web 服务方法的 PHP 函数。

（2）用 xmlrpc_server_create()函数创建 XML-RPC 服务器句柄。

（3）用 xmlrpc_server_register_method()函数将（1）中定义的 PHP 函数注册为 XML-RPC 方法。这些 PHP 函数即成为 Web 服务。

（4）利用 PHP 系统常量$HTTP_RAW_POST_DATA 获得 XML-RPC 请求数据，其值为 XML 文档，含有要执行的远程方法名和参数值。

（5）用 xmlrpc_server_call_method()函数接收并处理 XML-RPC 请求，调用 XML-RPC 服务器中指定的方法，返回该方法调用后的响应结果，响应结果为 XML 文档。

（6）输出响应结果，以便 XML-RPC 客户程序读取响应结果。

（7）用 xmlrpc_server_destroy()函数释放服务器资源。

【例 13.1】 以下 PHP 程序定义了两个用户自定义函数，并注册为 Web 服务。一个是计算两个数的和，另一个是根据客户端传递来的名字，返回字符串“XXXX，你好”。程序如下：

```
<?
/* 程序文件名:xmlrpc_server.php  */
function greet_func($method, $params, $app_data)
/* $method:Web 服务名;$params:请求的参数数组,$app_data: 应用程序参数*/
{
    $name = $params[0];
    return $name->scalar.",你好";
}

function sum_func($method, $params, $app_data)  //求两个数之和
{
   $num1=$params[0];
   $num2=$params[1];
   return $num1+$num2;
}
$server = xmlrpc_server_create();
$output_options = array("output_type" => "xml","encoding" => "gb2312");
xmlrpc_server_register_method($server, "greet", "greet_func");
xmlrpc_server_register_method($server, "sum", "sum_func");
$request_xml= $HTTP_RAW_POST_DATA;
$response=xmlrpc_server_call_method($server,$request_xml,null,$output_
options);
print $response;
xmlrpc_server_destroy($server);
?>
```

13.2.3 编写基于 PHP 5 和 XML-RPC 的 Web 客户端程序

XML-RPC 的 Web 客户端程序是向 Web 服务发送 POST 请求数据，获取响应结果的程序。在 PHP 5 中，为了实现 XML-RPC 的 Web 客户端与 Web 服务之间的数据通信，需要在两者之间建立 Socket 连接，然后通过该 Socket 连接，向 Web 服务发送 HTTP 的 POST 请求数据。最后等待 Web 服务返回的响应结果。与 Web 客户端程序相关的 XML-RPC 扩展函数如下。这些函数用于在 PHP 本地数据类型与请求或响应的 XML 编码数据之间的转换。

1. xmlrpc_encode_request()函数

格式：string xmlrpc_encode_request(string method, mixed params [, array output_options])

该函数生成方法调用或者应答的 XML 文档。执行成功返回值是 XML 字符串，否则返回 false。

参数说明：method 指定远程服务器上要调用的 Web 服务方法，如果为 null，则生成的 XML 是响应数据；params 是任何类型的参数数据，它应与远程服务器的方法的参数匹配；output_options 是可选的数组，用于指定编码类型。

2. xmlrpc_decode_request()函数

格式：array xmlrpc_decode_request(string xml, string& method [, encoding])

该函数将 XML 解码为本地的 PHP 数据类型，它也返回方法名。返回值是任何类型的值，通常为数组。

参数说明：xml 是要解码的原始 XML；method 是要存储方法名的变量，它以传递地址方式传递。如果不给出方法调用，其值不变。encoding 是输入编码，默认为 iso-8859-1。

3．xmlrpc_encode()函数

格式：string xmlrpc_encode(mixed value)

该函数将 PHP 的值编码为 XML。如果成功，返回值为 XML 字符串；否则，返回 false。value 参数是要被编码的 PHP 的值。

4．xmlrpc_decode()函数

格式：array xmlrpc_decode(string xml [,string encoding])

该函数将 XML 解码为本地的 PHP 数据类型。返回值是任何类型的数据，通常为数组。

参数说明：xml 为要解码的原始 XML。encoding 参数指定输入编码类型，默认为 iso-8859-1。

PHP 的 Web 客户端程序的编程步骤如下：

（1）Web 客户端与 Web 服务所在的主机建立 Socket 连接。

（2）向该 Socket 写入 XML 编码的 XML-RPC 请求数据。该请求数据按照 13.1 节介绍的 HTTP 报头和 XML-RPC 请求格式来发送。

（3）等待 Web 服务返回响应结果。从 Socket 中读取 Web 服务返回的响应结果，此响应为 XML 格式的字符串，需要将响应结果解析为 PHP 本身的数据类型。

（4）对返回值作进一步的处理。

上述（1）～（3）步是任何 XML-RPC 的 Web 客户端程序所必需的。因此，把这三步编写为自定义函数形式，存放到公共程序模块 rpcuntils.php，其他客户端程序在调用 rpcuntils.php 文件的函数之前，需用 include("rpcuntils.php")语句嵌入其代码。rpcuntils.php 程序如下：

```
<?php
/* 程序文件名: rpcuntils.php */
function send_http_post($request,$host,$uri,$port,$debug) {
    if ($host && $uri && $port) {
      $content_len = strlen($request);
      dbg("opening socket to host: $host, port: $port, uri: $uri", $debug);
      $fp = fsockopen($host, $port, $errno, $errstr);
      if ($fp) {
         $http_request = "POST $uri HTTP/1.0\r\n" .
           "User-Agent: xmlrpc-epi-php/1.0 (PHP)\r\n" .
           "Host: $host:$port\r\n" .
           "Content-Type: text/xml\r\n" .
           "Content-Length: $content_len\r\n\r\n" . $request;
         fputs($fp, $http_request, strlen($http_request));
          $response_buf = "";
```

```
            while (!feof($fp)) {
              $response_buf .= fgets($fp);
            }
            fclose($fp);
          }
          else
            dbg("打开 Socket 失败", $debug);
      }
      else
         dbg("主机、路径参数错误", $debug);
      return $response_buf;
   }

   function rpc_call($host,$uri,$port=80,$method,$args) {
     $output_options = array("output_type" => "xml","encoding" => "gb2312");
     $request_xml = xmlrpc_encode_request($method, $args, $output_options);
   $response_buf=send_http_post($request_xml, $host, $uri, $port, $debug);
     $xml = substr($response_buf, strpos($response_buf,"<?xml"));
     $return_val = xmlrpc_decode($xml);
     return $return_val;
   }
   function dbg($msg, $debug_level) {  //显示错误信息
      if ($debug_level >= 1) {
        echo "<h3>$msg</h3>"; flush();
      }
   }
   ?>
```

其中，send_http_post()函数向 Web 服务器的由$URI 变量指定的程序发送 POST 请求信息，然后等待 Web 服务返回的应答结果。rpc_call()函数把 XML-RPC 请求参数编码为 XML 格式，调用 send_http_post()函数，得到返回的 XML 格式应答结果，再把应答结果解码为 PHP 本身的数据类型。下面以两个例子来说明上述两个 Web 服务的调用过程。

【例 13.2】 调用 Web 服务 greet，显示字符串“徐辉，你好”。其中参数值“徐辉”由客户端传递给 Web 服务 greet。为测试方便，将服务程序 xmlrpc_server.php 存放到本地 Web 站点根目录下的 xmlrpc 子目录中（xmlrpc_client1.php）。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>XML-RPC 客户程序示例</title>
</head>
<body>
<?
include "rpcuntils.php";
$host = 'localhost';    //指定 Web 服务所在的主机、路径和端口
//$uri = '/xmlrpc/xmlrpc_server.php';
```

```
$uri = '/PHP5BOOK/ch13/xmlrpc/xmlrpc_server.php';
$port = 80;
$name="徐辉";
xmlrpc_set_type(&$name,"base64");
$args=array($name);
$method="greet";
$result = rpc_call($host,$uri,$port,$method,$args);
print $result."<br><br>";
?>
</body>
</html>
```

需要注意的是，在默认的情况下，字符串的编码为 iso8601 编码，对英文而言，在 Internet 上能够正确地传递和接收。但本例传递的参数值是中文，需要用 XML-RPC 扩展的 xmlrpc_set_type()函数将参数$name 的值转换为 base64 编码格式，按二进制数据传送，这样 Web 服务才能正确接收该参数。输出结果如图 13.4 所示。

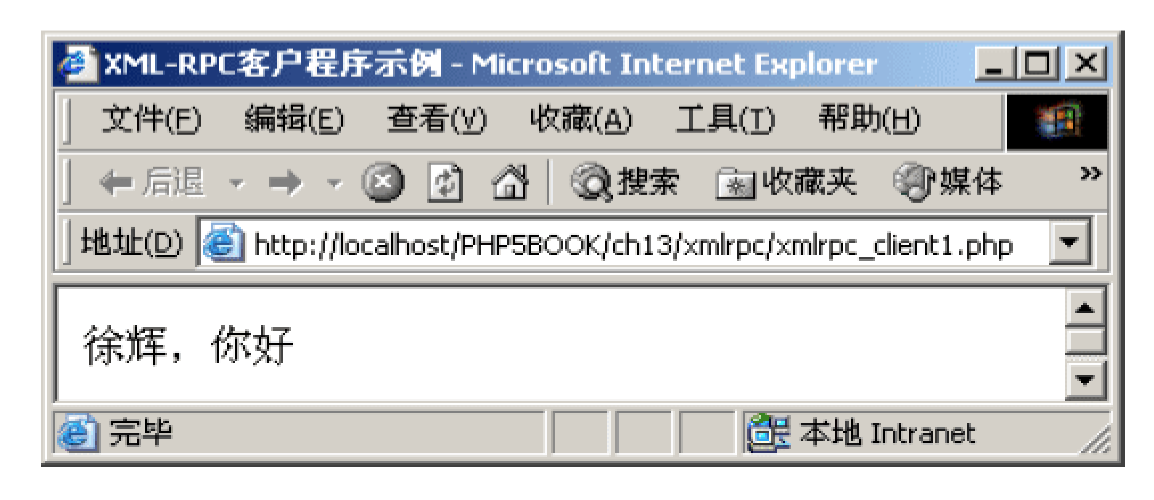


图 13.4 例 13.2 的输出结果

【例 13.3】 在客户端输入两个数值，调用 Web 服务 sum，计算这两个数的和，返回给客户端（xmlrpc_client2.php）。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>求两个数的和</title>
</head>
<body>
<?
include "rpcuntils.php";
$host = 'localhost';
$uri = '/PHP5BOOK/ch13/xmlrpc/xmlrpc_server.php';
$port = 80;
if ($_POST["n1"] && $_POST["n2"]) {
$n1=$_POST["n1"];
$n2=$_POST["n2"];
$args=array($n1,$n2);
$method="sum";
```

```
$result = rpc_call($host,$uri,$port,$method,$args);
print $n1."+".$n2."=".$result."<br>";
}
?>
<form name="form1" method="post" action="<? echo $_PHPSELF;?>">
  输入两个数值
  <input name="n1" type="text" id="n1" size="10">
  <input name="n2" type="text" id="n2" size="10">
  <input type="submit" name="Submit" value="提交">
</form>
</body>
</html>
```

13.3 SOAP 协议

除了利用 XML-RPC 协议实现 Web 服务的数据交换功能以外，还可以利用 SOAP 协议作为 Web 服务交换数据的协议。这一节介绍基于 SOAP 协议的 Web 服务所需要的一些协议及其原理，在 13.4 节将介绍如何利用 PHP 5 和 SOAP 实现 Web 服务。

13.3.1 SOAP 协议

简单对象访问协议（simple object access protocol，SOAP）是在分布式环境中通过 XML 编码进行分布式应用程序之间交换数据的网络协议，是一种轻量级的、基于 XML 的协议。从本质来说，SOAP 属于应用层协议，通常采用 HTTP 协议进行通信。SOAP 规定了 XML 消息所使用的格式、XML 消息的处理方法、一组应用程序定义的数据类型编码规则以及表示远程过程调用请求和应答的约定。它是平台无关的、编程语言无关的协议。这使得 SOAP 能够被用于从消息传递到 RPC（远程过程调用）的各种系统中。

一个 SOAP 消息是由一个必需的 SOAP 信封、一个可选的 SOAP 消息头和一个必需的 SOAP 消息体组成的 XML 文档。这三个组成部分如下：

（1）SOAP 信封。其元素名为<Envelope>，是表示 SOAP 消息的 XML 文档的顶级元素。

（2）可选的 SOAP 消息头。其元素名为<Header>。SOAP 消息头扩展了 SOAP 的功能，提供向 SOAP 消息中添加关于这个 SOAP 消息的某些特性的机制，如事务处理和安全性的支持。如果给出 SOAP 消息头，那么它必须是<Envelope>元素的第一个子元素。

（3）SOAP 消息体。它包含了调用请求和参数信息或者应答返回的数据。此外，SOAP 为 SOAP 消息体还定义了一个可选的<fault>元素，它是<Body>元素的子元素，用来描述错误代码和错误信息。

一个基本的 SOAP 消息的格式如下:

```
<soap:Envelope
   xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
   soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
    <soap:Header>
```

```
        ⋮(消息头内容)
    </soap:Header>
    <soap:Body>
        ⋮(消息体内容)
    </soap:Body>
</soap:Envelope>
```

所有信息都包括在<Envelope>元素中，其中，xmlns 属性定义了<Envelope>元素需要和 http://www.w3.org/2001/12/soap-envelope 命名空间相关联，该命名空间的前缀名定义为 soap。SOAP 消息头的直接子元素都必须限定命名空间前缀，SOAP 消息体的直接子元素可以指定命名空间前缀。encodingStyle 属性用于指明在 SOAP 消息中使用哪种编序规则，该属性的作用范围包括该元素的内容和所有其子元素中未使用该属性的所有子元素。

例如，假设某一 Web 网站 www.example.com 提供一个根据邮政编码查询天气预报的 Web 服务，该 Web 服务方法名为 getWeatherForCode。为了获取某一邮政编码所在地区的天气预报，可以发送一个 SOAP 消息，请求调用 Web 服务的 getWeatherForCode 方法，向该方法传递一个邮政编码，SOAP 消息内容如下：

```
<soap:Envelope
   xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
   soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:Body>
  <wi:getWeatherForCode
      xmlns:wi='http://www.example.com/WeatherService'>
    <wi:ZipCode>94121</wi:ZipCode>
  </wi:getWeatherForCode>
</soap:Body>
</soap:Envelope>
```

位于网站 www.example.com/WeatherService 的 Web 服务收到 SOAP 请求后，执行 getWeatherForCode 方法，返回的 SOAP 响应消息大体如下：

```
<soap:Body>
  <wi:getWeatherForCodeResponse
      xmlns:wi='http://www.example.com/WeatherService'>
     <wi:result>59F (15C)</wi:result>
  </wi:getWeatherForCodeResponse>
</soap:Body>
```

关于 SOAP 消息的详细内容参见网站 http://www.w3.org/TR/2003/REC-soap12-part0-20030624/。

13.3.2　WSDL 协议

SOAP 协议标准定义了如何对发送到 Web 服务的消息进行格式化和编码。但是，如何知道给服务器发送了什么消息？一个 Web 服务提供了哪些方法？这些方法应该接收哪些

参数、返回什么类型的值呢？对于这些问题，较好的解决方案是使用 WSDL，称之为 Web 服务描述语言（web services description language）。

WSDL 是一个用于描述 Web 服务所提供的方法以及这些方法的访问方式的 XML 文档。在一个 WSDL 文档中，所有的参数和 Web 服务的方法名称都有详细的说明，同时还包括了 Web 服务的位置、绑定的传输协议等。利用 WSDL，可以生成具体的 Web 服务所必需的有效 SOAP 消息，并且该 SOAP 消息也被发送到该 Web 服务。

一个 WSDL 文档是由五个主要元素组成的 XML 文档，这五个元素的描述归纳如下：

- types 元素。types 元素定义了在 WSDL 文档中使用的自定义数据型。WSDL 本身没有专用的数据类型，但它使用 W3C 的 XML Schema 规范作为默认的数据类型。如果所描述的服务仅用到 XML Schema 内置的一些简单类型，如 string、integer 等，那么可以省略 types 元素。
- message 元素。message 元素定义服务接收和返回的通信消息的数据结构。它们使用 types 所定义的类型来定义整个消息的数据结构。
- portType 元素。portType 元素是服务所支持的抽象操作的列表，抽象操作主要是对 Web 服务的方法名进行转换，此外它还定义了所有操作接收和返回的逻辑消息。在 portType 里能够定义多个操作。
- binding 元素。binding 元素描述如何在 Internet 上实现服务的具体细节，包括对每个端口定义其消息格式和传输的协议。例如，SOAP 协议就是在通信中 Web 服务能够解析和识别的协议。
- service 元素。service 元素定义了调用 Web 服务的地址。通常 service 元素包含一个用于调用 SOAP 服务的 URL 和服务的描述信息。

上述五个元素必须包含在 WSDL 文档的一个<definitions>元素中。<definitions>元素格式为：

```
<?xml version="1.0"?>
<definitions  name ="weather"
  targetNamespace="http://www.example.com/WeatherService/weather.wsdl"
  xmlns:tns="http:// www.example.com/WeatherService/weather.wsdl"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://schemas.xmlsoap.org/wsdl/">
```

其中，name 属性表示 WSDL 文件名称，此外，还定义了文档中的多个命名空间前缀，这些命名空间前缀使文档中的每个元素都具有唯一性，每个被限定的元素都可以引用正确的命名空间。

【例 13.4】 虚构的天气预报 Web 服务的 WSDL 文档内容。

```
<?xml version="1.0"?>
<definitions  name ="weather"
  targetNamespace="http://www.example.com/WeatherService/weather.wsdl"
  xmlns:tns="http:// www.example.com/WeatherService/weather.wsdl"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
```

```
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://schemas.xmlsoap.org/wsdl/">
  <!—
定义一个请求消息,一个响应消息
-->
<message name='getWeatherRequest'>
  <part name='zipcode' type='xsd:string'/>
</message>
<message name='getWeatherResponse'>
  <part name='result' type='xsd:string'/>
</message>
  <!—
定义一个端口类型,在其中定义一个方法 getWeather
-->
<portType name="GetWeatherPortType">
   <operation name="getWeather">
     <input message="getWeatherRequest"/>
     <output message="getWeatherResponse"/>
   </operation>
</portType>
<!—
   设置绑定,通过 HTTP 上的 SOAP 协议访问端口类型,用 RPC 来调用服务
 -->
<binding name="GetWeatherBinding" type="GetWeather">
  <soap:binding style="rpc"
    transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="getWeather">
       <soap:operation soapAction="getWeather"/>
       <input>
         <soap:body
           encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
           namespace="http://www.example.com/WeatherService"
           use="encoded"/>
       </input>
       <output>
          <soap:body
           encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
           namespace=" "
           use="encoded"/>
        </output>
   </operation>
  </binding>
```

```
        <!--
              给出服务的地址,它以 SOAP 协议为基础
         -->
        <service name="GetWeatherService">
          <documentation>使用本服务获取某一地区的天气预报</documentation>
          <port name="GetWeather" binding="GetWeatherBinding">
            <soap:address
            location="http://www.example.com:80/WeatherService"/>
          </port>
        </service>
      </definitions>
```

下面结合此 WSDL 文档实例，对构成 WSDL 的五个主要元素作简单的介绍。

1．types 元素

WSDL 文档的 types 元素包含了 Web 服务所利用的所有具体的或者复合的数据类型。如果没有需要声明的数据类型，可以省略此元素。

2．message 元素

WSDL 文档的 message 元素描述 Web 服务接收和返回的抽象消息。这些消息包括方法调用的输入参数消息和返回的响应输出参数消息。例如：

```
<message name='getWeatherRequest'>
  <part name='zipcode' type='xsd:string'/>
</message>
<message name='getWeatherResponse'>
  <part name='result' type='xsd:string'/>
</message>
```

上述代码定义了两个消息：第 1 个消息名为 getWeatherRequest，后缀 Request 表示该消息定义的是 getWeather 方法的输入参数，为字符串型的输入参数 zipcode。第 2 个消息名为 getWeatherResponse，后缀 Response 表示定义该方法的输出参数，为一个字符串型的参数 result。

如果方法有多个参数，那么 message 元素包含多个 part 元素。每个 part 元素都有名字和属性。name 属性指定参数名，type 属性给参数指定一个 XML Schema 的数据类型。type 属性的值必须是 XML Schema 有效名称，例如，zipcode 的 type 属性值设置为 xsd:string，表示字符串类型。

3．portType 元素

portType 元素定义一组操作，这些操作构成 Web 服务所提供的方法。每个操作由一个单独的输入消息和一个单独的输出消息组成。

```
<portType name="GetWeatherPortType">
    <operation name="getWeather">
      <input message="getWeatherRequest"/>
      <output message="getWeatherResponse"/>
```

```
    </operation>
</portType>
```

一个<portType>元素内可以包含一个或多个<operation>元素，每个<operation>元素定义一个操作。在<operation>元素中，可以包含有<input>、<output>和<fault>元素，每个<input>和<output>元素的消息引用已定义的<message>元素。

<portType>元素声明了一个名为 getWeather 操作，该操作的功能是接收和返回一个消息，这两个消息已经在前面的 message 元素中定义过，分别由<input>元素和<output>元素引用。

4. binding 元素

<binding>元素为 WSDL 文档中的 portType 元素的每个操作定义如何在网络中实际传输的具体细节。WSDL 可以与多种传输方式绑定，包括 HTTP GET、HTTP POST 或 SOAP。对每一个端口类型也可以指定多种绑定。

在例 13.4 中，binding 元素的内容以下：

```
<binding name="GetWeatherBinding" type="GetWeather">
  <soap:binding style="rpc"
      transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="getWeather">
      <soap:operation soapAction="getWeather"/>
      <input>
        <soap:body
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace="http://www.example.com/WeatherService"
          use="encoded"/>
      </input>
      <output>
        <soap:body
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace=" "
          use="encoded"/>
      </output>
  </operation>
</binding>
```

可以看到，binding 元素的 type 属性引用了该绑定正在定义的 portType 元素。binding 元素内有两个子元素。

（1）soap:binding 元素。该元素告诉我们使用 HTTP 协议来发送 SOAP 消息，style 属性的值决定了被发送的 SOAP 消息的格式，值 rpc 表示用 XML-RPC 来表示 SOAP 消息。

（2）operation 元素。该元素引用了 portType 元素的一个操作。<soap:operation>元素定义当把消息发送给 Web 服务时 soapAction HTTP 标题的内容。在<input>和<output>元素内，指定前面被定义的消息必须被编码，并作为 SOAP 消息而被传递。

5. Service 元素

WSDL 文档的最后一个元素是<Service>元素。<Service>元素包含一个或多个<port>元

素，每个<port>元素指定一个 Web 服务的方法所在主机地址。在例 13.4 的 service 元素中，使用<soap:address>元素指定 Web 服务的主机地址为 http://www.example.com:80/WeatherService。

13.4 基于 PHP 5 和 SOAP 的 Web 服务

13.4.1 支持 SOAP 的 PHP 5 配置

为了在 PHP 5 中使用基于 SOAP 的 Web 服务，需要在 PHP 5 中添加 SOAP 扩展模块。根据系统平台环境的不同，配置方法有所不同。

在 UNIX 和 Linux 系统中，为了支持 SOAP 协议，必须在配置和编译 PHP 时，带上开关参数--enable-soap。这样，编译后的 PHP 才支持 SOAP 协议，从而可以使用 Web 服务。如果编译时不带此开关参数，就不能使用 SOAP 的类或函数，因而不能调用 Web 服务。

对于 Windows 系统，要在 PHP 5 中支持 SOAP，只需在 PHP 的配置文件 php.ini 中增加一行：extension=php_soap.dll。

在 PHP 5 的 php.ini 配置文件中，有三个与 SOAP Web 服务相关的配置选项，它们是与 WSDL 文件处理相关的配置指令。

（1）soap.wsdl_cache_enabled：设置是否缓存 WSDL 文档。其默认值为 on，它告诉 PHP 缓存 WSDL 文件。如果设置为 off，则每次创建 Web 服务的连接时都下载 WSDL 文件。

（2）soap.wsdl_cache_dir ：设置 WSDL 文件的缓存目录。其默认值为/tmp。注意，在 Windows 系统中要更改此指令的值。

（3）soap.wsdl_cache_ttl：设置 WSDL 文件的缓存时间（秒）。默认值为 86 400 秒（即 1 天）。

13.4.2 基于 SOAP 的 Web 服务的客户端程序设计

在 PHP 5 中配置好 SOAP 扩展后，使用 SOAP 扩展的 SoapServer 类和 SoapClient 类所提供的方法，编写 Web 服务及其客户端程序。

SoapClient 类隐藏了 SOAP 协议和 WSDL 协议的大部分细节，使得软件开发人员利用它提供的一组方法，可以很容易地设计出 Web 服务的客户程序。Web 服务的客户程序的设计过程为：首先用 SoapClient 类的构造方法 SoapClient()创建一个 SoapClient 对象，此对象建立了与指定 Web 服务的调用机制。然后通过调用 SoapClient 对象的方法，调用 Web 服务的方法，返回一个响应结果。SoapClient 类的常用方法如下。

1．SoapClient()构造方法

格式：object SoapClient(mixed wsdl_uri [,array options])

SoapClient()构造方法以 WSDL 模式或非 WSDL 模式实例化一个 SoapClient 类的对象。

wsdl_uri 参数指定与 Web 服务相应的 WSDL 文件所在的 URI 地址。如果不通过 WSDL 文档指定服务（即非 WSDL 模式），则 wsdl_uri 参数设置为 NULL，此时由 options 参数指定 Web 服务的地址。如果是 WSDL 模式，则 options 参数是可选。

当采用非 WSDL 模式时，options 参数数组必须设置 location 和 uri 参数。options 数组的参数如下所示。

- location：请求的 Web 服务的 URL 地址。
- uri：指定 Web 服务的名字空间。
- login：如果使用 HTTP 认证来访问 SOAP Web 服务，此参数用来指定用户名。
- password：如果使用 HTTP 认证来访问 SOAP Web 服务，此参数指定用户密码。
- proxy_host：当通过代理服务器与 Web 服务连接时，此参数指定代理主机名。
- proxy_login：指定代理服务器的用户名。
- proxy_password：指定代理服务器的用户密码。
- proxy_port：指定代理服务器的端口号。
- soap_version：指定 SOAP 版本是 SOAP 1.1 还是 SOAP 1.2。默认值是 1.1 版本，其值写为 SOAP_1_1。如果是 SOAP 1.2，其值写为 SOAP_1_2。
- trace：如果想查看 SOAP 请求和响应消息，需要将此参数的值设置为 1。

下面以一个实际例子，说明如何访问现有的 Web 服务。在 http://www.xmethods.com 网站上有许多 Web 服务示例，其中有一个称为“Currency Exchange Rate”的 Web 服务，调用该服务可以查看各国货币之间的兑换比率。该服务的 WSDL 文件的地址为 http://www.xmethods.net/sd/2001/CurrencyExchangeService.wsdl。

下面的代码利用该 WSDL 文件建立一个 SoapClient 对象，此对象含有连接到 xmethods.net 的“Currency Exchange Rate”服务的信息。

```
<?php
$wsdl="http://www.xmethods.net/sd/2001/CurrencyExchangeService.wsdl";
$client = new SoapClient($wsdl);
?>
```

到目前为止，还不知道“Currency Exchange Rate”服务提供哪些方法。为了查看该服务的方法，可以在浏览器打开该 WSDL 文件，从中查到 Web 服务的方法。当然，也可以用 SoapClient 类_ _getFunctions()方法来查看该服务所提供的方法。

2．_ _getFunctions()方法

格式：array SoapClient->_ _getFunctions()

该方法返回一个由 SoapClient 对象调用的服务中所包含的所有方法构成的字符串数组。

下面的代码获取 xmethods.net 上的“Currency Exchange Rate”服务的可用方法列表。

```
<?php
$wsdl="http://www.xmethods.net/sd/2001/CurrencyExchangeService.wsdl";
$client = new SoapClient($wsdl);
var_dump($client->__getFunctions());
?>
```

该程序的返回值如下：

```
array(1) {
  [0]=>string(49) "float getRate(string $country1, string $country2)"
}
```

从这一返回结果知道，“Currency Exchange Rate”服务提供一个 getRate()方法，它接受两个输入参数：两个国家名字符串，返回值的类型是字符串型。下面给出一个调用该 Web 服务的方法的客户程序。

创建了 SoapClient 对象后，就可以调用 Web 服务的方法。SoapClient 类的聪明之处在于它自动地把 Web 服务的方法作为自身对象的方法来调用。例如，从上述代码的输出结果，知道Web 服务的 getRate 方法及其参数形式，因此，只需调用已创建的 SoapClient 对象的 getRate 同名方法，即可调用远程服务器的“Currency Exchange Rate”服务。

【例 13.5】 以 WSDL 模式调用 www.xmethods.net 上“Currency Exchange Rate”服务的 getrate 方法的客户端程序（ex13_5.php）。

```
<?php
$wsdl="http://www.xmethods.net/sd/2001/CurrencyExchangeService.wsdl";
try {
  $client = new SoapClient($wsdl);  //根据 WSDL 建立客户端对象
  // 调用 getRate()方法
  $exchange = $client->getRate('united states', 'china');
  echo "1 美元兑换为".$exchange."元人民币 ";
}
catch (SoapFault $sf) {
  echo "<p>Currency Exchange Rate 服务出现故障,稍后再访问</p>";
  echo "错误代码: ".$sf->faultcode;
  echo "<br/>错误信息: ".$sf->faultstring;
  exit;
}
?>
```

在本程序中，使用 try 语句对执行 SoapClient 对象的方法期间产生的异常进行处理。SoapFault 类将错误代码和错误信息分别存放到其两个成员变量 faultcode 和 faultstring。因此，在 try 语句的 catch 块中，除了处理其他异常外，还要处理 SoapFault 异常。访问该程序，输出的结果如下：

```
1 美元兑换为 7.824 元人民币
```

如果不指定 WSDL 文件的地址，则使用非 WSDL 模式来调用 Web 服务。在 http://www.xmethods.net 的“Currency Exchange Rate”服务页面上，单击超链接“View RPC Profile.”，将看到从 WSDL 文件提取的内容，用于手工配置 SOAP RPC 调用，其内容如表 13.2 所示。

表 13.2　Web 服务的 RPC 参数

方法名	getRate	
URL	http://services.xmethods.net:80/soap	
SOAPAction		
方法的命名空间 URI	urn:xmethods-CurrencyExchange	
输入参数及其类型	country1	string
	country2	string
输出参数及类型	Result	float

【例 13.6】 以非 WSDL 模式调用 www.xmethods.net 上“Currency Exchange Rate”服务的客户端程序（ex13_6.php）。

```
<?php
 $client = new SoapClient(NULL,
     array("location" => "http://services.xmethods.net:80/soap",
          "uri"    => "urn:xmethods-CurrencyExchange",
          "style"  => SOAP_RPC,
          "use"    => SOAP_ENCODED
    ));
  $x=$client->__soapCall("getRate",array('united states','china'));
  echo "1 美元兑换为".$x."元人民币 ";
?>
```

3．__getLastRequest()方法

格式：string SoapClient->__getLastRequest()

该方法用于查看 SOAP 请求消息。如果要查看 SOAP 请求和响应的内容，在创建 SoapClient 对象时，需要在 SoapClient 类的构造方法中打开 tracing 参数，如下：

```
$client = new SoapClient($wsdl,array('trace' => 1));
```

其中，构造方法的第 2 个参数是可选的配置值的数组。通过指定 trace 选项的值，可以查看 SOAP 请求和响应信息。在执行 SOAP 请求后，调用__getLastRequest()方法，获得 SOAP 请求内容。

【例 13.7】 查看 Currency Exchange Rate 服务的 SOAP 请求消息（ex13_7.php）。

```
<?php
$wsdl="http://www.xmethods.net/sd/2001/CurrencyExchangeService.wsdl";
$client = new SoapClient($wsdl,array('trace' => 1));
$exchange = $client->getRate('united states', 'china');
echo htmlspecialchars($client->__getLastRequest());
?>
```

该程序的返回值如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
        xmlns:SOAP-ENV="http:  //schemas.xmlsoap.org/soap/envelope/"
        xmlns:ns1="urn:xmethods-CurrencyExchange"
        xmlns:xsd="http:       //www.w3.org/2001/XMLSchema"
        xmlns:xsi="http:       //www.w3.org/2001/XMLSchema-instance"
        xmlns:SOAP-ENC="http:  //schemas.xmlsoap.org/soap/encoding/"
        SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
        <ns1:getRate>
```

```
            <country1 xsi:type="xsd:string">united states</country1>
          <country2 si:type="xsd:string">china</country2>
      </ns1:getRate>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

4. _ _getLastResponse()方法

格式：object SoapClient->_ _getLastResponse()

该方法用于查看调用 Web 服务后的 SOAP 响应消息。它与_ _getLastRequest()方法一样，使用前要打开 tracing 选项。

【例 13.8】 查看 Currency Exchange Rate 服务的 SOAP 响应消息（ex13_8.php）。

```
<?php
$wsdl="http://www.xmethods.net/sd/2001/CurrencyExchangeService.wsdl";
$client = new SoapClient($wsdl,array('trace' => 1));
$exchange = $client->getRate('united states', 'china');
echo htmlspecialchars($client->__getLastRequest());
?>
```

该程序的输出结果如下（为便于阅读而格式化过输出结果）：

```
<?xml version='1.0' encoding='UTF-8'?>
<soap:Envelope xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/'
    xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
    xmlns:xsd='http://www.w3.org/2001/XMLSchema'
    xmlns:soapenc='http://schemas.xmlsoap.org/soap/encoding/'
    soap:encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'>
<soap:Body>
<n:getRateResponse xmlns:n='urn:xmethods-CurrencyExchange'>
          <Result xsi:type='xsd:float'>7.824</Result>
      </n:getRateResponse>
   </soap:Body>
</soap:Envelope>
```

13.4.3 基于 SOAP 的 Web 服务程序设计

SOAP 扩展的 SoapServer 类提供了编写 Web 服务的几个方法，为了说明如何编写 Web 服务端程序，下面以一个简单的例子来介绍其编程过程。首先给出该 Web 服务的 WSDL 文档内容，如例 13.9 所示。

【例 13.9】 一个提供数学计算服务的 WSDL 文件（math.wsdl）。

```
<?xml version="1.0"?>
<definitions
    xmlns="http:              //schemas.xmlsoap.org/wsdl/"
    xmlns:xsd="http:          //www.w3.org/2001/XMLSchema"
    xmlns:tns="http:          //example.com/PHP5BOOK/ch13/soap/math.php"
    xmlns:soap-env="http:     //schemas.xmlsoap.org/wsdl/soap/"
```

```
    xmlns:wsdl="http:        //schemas.xmlsoap.org/wsdl/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    targetNamespace="http:  //example.com/PHP5BOOK/ch13/soap/math.php">
<message name="mInput">
    <part name="a" type="xsd:float"/>
    <part name="b" type="xsd:float"/>
</message>
<message name="mOutput">
    <part name="return" type="xsd:float"/>
</message>
<portType name="MathServicePortType">
    <operation name="add">
        <input message="tns:mInput"/>
        <output message="tns:mOutput"/>
    </operation>
    <operation name="divide">
        <input message="tns:mInput"/>
        <output message="tns:mOutput"/>
    </operation>
</portType>
<binding name="MathServiceBinding" type="tns:MathServicePortType">
  <soap-env:binding xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
        style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation xmlns:default="http://schemas.xmlsoap.org/wsdl/soap/"
                  name="add">
            <input xmlns:default="http://schemas.xmlsoap.org/wsdl/soap/">
              <soap-env:body xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
                  use="encoded"
  encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
            </input>
            <output xmlns:default="http://schemas.xmlsoap.org/wsdl/soap/">
               <soap-env:body xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
                  use="encoded"
  encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
          </output>
      </operation>
      <operation xmlns:default="http://schemas.xmlsoap.org/wsdl/soap/"
                  name="divide">
      <input xmlns:default="http://schemas.xmlsoap.org/wsdl/soap/">
          <soap-env:body xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
               use="encoded"
    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
        </input>
        <output xmlns:default="http://schemas.xmlsoap.org/wsdl/soap/">
            <soap-env:body xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
```

```
                    use="encoded"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
            </output>
        </operation>
    </binding>
    <service name="MathServiceService">
        <port xmlns:default="http://schemas.xmlsoap.org/wsdl/soap/"
              name="MathServicePort"
              binding="tns:MathServiceBinding">
        <soap-env:address xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
              location="http://localhost/PHP5BOOK/ch13/soap/math.php"/>
        </port>
    </service>
    </definitions>
```

该 WSDL 文档描述了 Web 服务的两个操作方法：add(a,b)和 divide(a,b)，分别表示两个数的和、商。下面介绍与 Web 服务相关的 SoapServer 类的方法，然后用这些方法编写实现这两个操作的 Web 服务端程序。

1．SoapServer()构造方法

格式：object SoapServer(mixed wsdl[, array options])

SoapServer()构造方法以 WSDL 模式或非 WSDL 模式来实例化 SoapServer 类的对象。如果采用 WSDL 模式，需要将 wsdl 参数的值设置为一个 WSDL 文件的地址，否则设置为 NULL。options 参数是一个数组，用来设置选项的值。

- actor: 定义 SOAP Web 服务的 URI。
- soap_version: 设置 SOAP 版本，必须按 SOAP_x_y 格式来设置，其中，x 是主版本号，y 是次版本号。例如，对于 SOAP 1.2 版，其值设置为 SOAP_1_2。

以下代码创建一个 SoapServer 对象，引用 math.wsdl 文件。

```
$soapserver = new SoapServer("math.wsdl");
```

如果 WSDL 文件存放在另一个服务器，那么，需要用 URL 来引用该服务器。例如：

```
$soapserver = new SoapServer("http://www.example.com/math.wsdl");
```

2．addFunction()方法

格式：void SoapServer->addFunction(mixed functions)

利用 addFunction()方法，将一个或多个 PHP 自定义函数设置为 Web 服务的方法，这些方法供 Web 服务客户端调用。functions 参数为对外公开的 PHP 自定义函数名。例如：

```
$soapserver->addFunction("add");
$soapserver->addFunction("divide");
```

当 WSDL 文件中定义多个函数时，通过给该方法传递一个以函数名为数组元素的数组，可以为 Web 服务的客户端公开多个方法，供 SOAP 客户端调用，例如，上面两条语句可以写为如下一条语句。

```
$soapserver->addFunction(array("add","divide");
```

如果要把所有自定义函数对外公开为 Web 服务的方法，可以给该方法的 functions 参数设置为 SOAP_FUNCTIONS_ALL 常量，比如：

```
$soapserver->addFunction(SOAP_FUNCTIONS_ALL);
```

3．handle()

格式：void SoapServer->handle([string soap_request])

该方法接收来自 Web 服务客户端的 SOAP 请求，调用相应的方法，返回响应结果给客户端。soap_request 参数存放 SOAP 请求消息，如果省略该参数，则 SOAP 请求存放到 PHP 的$HTTP_RAW_POST_DATA 全局变量。例如：

```
$soapserver->handle();
```

编写 Web 服务程序的基本步骤为：

（1）编写作为 Web 服务的 PHP 自定义函数；

（2）调用 SoapServer()构造方法创建一个 SoapServer 类对象；

（3）调用 addFunction()方法把自定义函数设置为 Web 服务；

（4）调用 handle()方法接收客户端的 SOAP 请求，调用相应的方法，返回响应结果。

【例 13.10】 提供数学计算服务的 Web 服务程序（math.php）。

```
<?php
function add($a, $b) { return $a + $b;}
function divide($a, $b) { return $a / $b;}
$soapserver = new SoapServer("math.wsdl");
$soapserver->addFunction(array("add","divide"));
$soapserver->handle();
?>
```

利用 addfunction()方法，可以将一个或多个 PHP 自定义函数设置为 Web 服务的方法。如果要将一个类的所有方法都设置为 Web 服务的方法，就不能用 addfunction()方法，如何解决呢？解决办法是采用下面的 setClass()方法。

4．setClass()方法

格式：void SoapServer->setClass(string class_name [, mixed args])

该方法将指定类中的方法设置为 Web 服务。class_name 参数指定类名，可选的 args 参数指定传递给类构造方法的任何参数。

例 13.10 的服务程序也可以编写为一个类，再用 setClass()方法将该类的方法设置 Web 服务方法，程序如下：

```
<?php
class MathService {      // 将 Web 服务创建为一个类
   function add($a, $b) { return $a + $b;}
   function divide($a, $b) { return $a / $b;}
```

```
}
$ss = new SoapServer("http://localhost/PHP5BOOK/ch13/soap/math.wsdl");
$ss->setClass('MathService');
$ss->handle();
?>
```

用 setClass()代替 addFunction()与任何客户端 SOAP 请求无关，客户端程序无须修改。下面给出调用上述 Web 服务的客户端程序。

【例 13.11】 调用 math.php 的 Web 服务的客户端程序（mathclient.php）。

```
<?php
$sc = new SoapClient("http://localhost/PHP5BOOK/ch13/soap/math.wsdl");
$answer = $sc->add(3, 4);
echo "3 + 4= {$answer}<br />\n";
$answer = $sc->divide(56, 7);
echo "56 / 7 ={$answer}<br />\n";
?>
```

访问该客户程序，输出结果如图 13.5 所示。

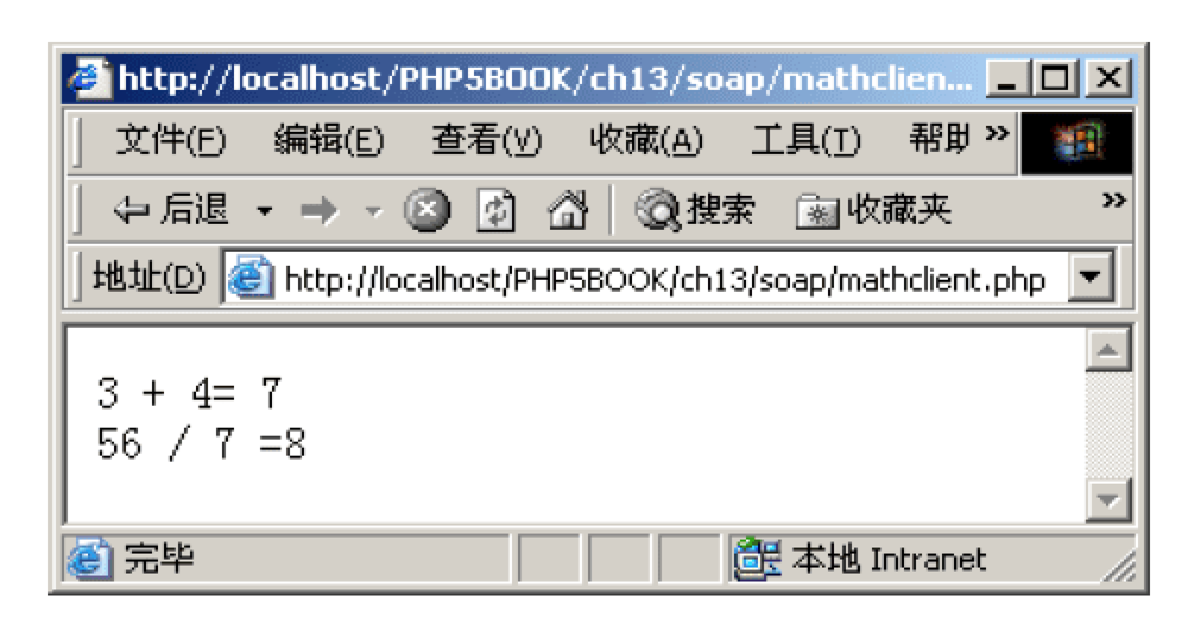


图 13.5 调用 Web 服务的输出结果

13.4.4 基于 PHP 5 和 Google Web API 的搜索引擎程序设计

Google 是一个非常优秀的搜索引擎。2002 年 4 月，Google 公司发布了受限的 Web 搜索服务 APIs，Google Web APIs 是一种 Web 服务，允许开发者将它的搜索引擎服务整合到自己的页面或者应用程序中，不需要通过它的网站而直接访问其搜索数据库。对于用户来说，感到这个搜索引擎是开发者提供的。另外，也使开发者对搜索引擎进行二次开发，如增加信息过滤、约束条件等功能。这里再介绍一个实例，将 PHP 5 和 Google Web 服务整合，介绍如何编写 PHP 的搜索引擎程序。

Google 搜索引擎提供了基于 SOAP 的 Web 服务。它允许开发者使用任何支持 Web 服务的编程语言，在不同的开发环境下编写客户端程序，调用其 Web 服务。这使得开发一套支持 Google 搜索引擎功能的应用程序变得非常容易。要使用 Google Web 服务 API，首先必须在 http://www.google.com/apis/网站上注册一个免费的 Google 账号，然后 Google 向你注册时提供的 E_mail 信箱发送一封确认信，确认后就会收到 Google 发送给你的 32 位字符

长度的唯一许可授权码。每次请求调用 Google Web 服务时，必须提供该许可授权码。

目前，Google Web 服务 API 处于 Beta 测试阶段，因此在使用上受到一些限制：每个许可授权码每天最多只能发送 1000 次查询；每次查询最多只能返回 10 条查询结果；每次查询，即使 Google 服务器上面的符合查询条件的结果超过 1000 条，最多只能返回前 1000 条查询结果中的 10 条结果；前 1000 条查询结果之后的查询结果不再返回给查询者。

Google Web 服务提供了以下三种方法。

1．doGetCachedPage()方法

格式：doGetCachedPage(string $key, string $url)

该方法从 Google 缓冲页面中检索与 URL 相符的页面内容。要注意的是用户使用该方法时看不到该 URL 更新后的页面内容。其中，$key 为许可授权码，$url 是页面的 URL。

2．doSpellingSuggestion()方法

格式：doSpellingSuggestion(string $key, string $phrase)

该方法对用户提交的搜索关键字进行拼写检查。如果拼写错误，或者引擎根本不认识某个单词，那么它将返回空值，否则就会返回所建议的词或词组。其中，$phrase 是要搜索的关键字。

3．doGoogleSearch()方法

格式：doGoogleSearch(string $key, string $q, int $start, int $maxResults,
boolean $filter, string $restrict, boolean $safeSearch,
string $lr, string $ie, string $oe)

该方法返回与用户输入的搜索关键字相符的查询结果。

其参数含义为：

- $key 是申请的许可授权码。
- $q 是要搜索的内容。
- $start 是返回的第一个查询结果在所有查询结果中的索引号（从 0 开始）。
- $maxResults 是一次返回的查询结果数，取值范围是 1～10。
- $filter 是网页过滤器，其值为布尔型，如果启用了过滤器，返回的结果集中只会包含来自同一主机或站点的前两个结果。
- $restrict 用于指定在某个国家中搜索网页，其值为国家代码，例如“coutryCN”表示中国。
- $safeSearch 是一个布尔值，在查询结果中是否过滤成人内容。
- $lr 对搜索的语言进行限制，例如“lang_zh-CN |lang_zh-TW”表示在简体中文和繁体中文网站中搜索。
- $ie 设置输入的编码
- $oe 设置输出的编码，$ie 和$oe 通常为 UTF-8 编码。

DoGoogleSearch()方法的返回值是一个 GoogleSearchResult 对象的实例，该对象的主要成员有：

- estimatedTotalResultsCount 表示搜索结果的总个数。
- resultElements 结果对象数组。
- searchQuery 搜索内容。

- startIndex　搜索结果的开始索引号。
- endIndex　搜索结果的结束索引号。
- searchTime　搜索用时。

其中，resultElements 成员是 ResultElement 对象类型的数组，该对象的主要成员有 URL、snippet、title，分别表示含有搜索内容的页面 URL、页面摘要和页面标题。

Google 为其 Web 服务提供了一个开发包，可以从其网站下载 googleapi.zip 压缩包，它包括了 Google Web API 使用文档、GoogleSearch.wsdl 文件以及 Java 和 C#两种编程语言 API 接口。GoogleSearch.wsdl 文件描述了 Google Web 服务所包含的三种方法的调用接口。在 PHP 程序中调用 Google Web 服务的关键技术如下：

（1）利用 SoapClient 类的构造方法，建立与 Google Web 服务的连接，返回 SoapClient 对象。代码为：

```
$sc = @new SoapClient('GoogleSearch.wsdl');
```

说明：将下载的 GoogleSearch.wsdl 文件复制到本地与 PHP 程序所在的同一个 Web 目录中。

（2）将 SOAP 形式描述的服务请求发送给 Google Web 服务，远程调用其 doGoogleSearch()方法，Google Web 服务将搜索结果又以 SOAP 形式通过 HTTP 返回给 PHP 客户程序。代码为：

```
$results = @$client->doGoogleSearch(MyKEY,trim($in_keywords),$start,
RESULTS_PER_PAGE, FALSE, '', FALSE, "",'','');
```

（3）输出返回值$results 的有关成员的值。

根据上述关键技术，编写了一个 PHP 程序文件 google_keywords.inc.php，它定义了一个 GoogleKeywords 类，其中公共、静态的 findAndPrintRelatedPages 方法搜索输入的关键字，并从搜索结果的指定索引号开始输出 10 条搜索结果。程序内容见例 13.12。

【例 13.12】 搜索引擎子程序（google_keywords.inc.php）。

```
<?
define('MyKEY', 'XXXXXXXXXXXXXXXX');    //申请的许可授权号
define('RESULTS_PER_PAGE', 10);         //每次输出搜索结果的数量
class GoogleKeywords
{
  public static function findAndPrintRelatedPages($in_keywords,$start)
  {
    try
    {
      $client = @new SoapClient('GoogleSearch.wsdl');
      $results = @$client->doGoogleSearch(MyKEY,trim($in_keywords),$start,
RESULTS_PER_PAGE, FALSE, '', FALSE, "",'','')
      self::emitSearchSummary($results);  //输出搜索摘要结果
      foreach ($results->resultElements as $resultObject) // 输出结果
        self::emitSearchResult($resultObject);
    }
```

```
    catch (SoapFault $sf)  {
      echo "SOAP 故障: {$sf->faultstring}<br/>\n";
    }
    catch (Exception $e)  {
      echo "异常: {$sf->faultstring}<br/>\n";
    }
  }
private static function emitSearchSummary($in_results) {
  echo <<<EOHEADER
 <TABLE cellSpacing=0 cellPadding=0 width="98%" bgColor=#e5ecf9 border=0>
  <TR>
     <TD>搜索到约有 <em>$in_results->estimatedTotalResultsCount</em>
项网页符合$in_results->searchQuery 的查询结果,以下是第$in_results->startIndex
- $in_results->endIndex 项.(搜索用时 $in_results->searchTime 秒)
</TD></TR></TABLE> <br/>
EOHEADER;
  }

  private static function emitSearchResult($in_result)  {
      echo <<<EORESULT
  <table width='100%' border='0' cellspacing='0' cellpadding='0'>
  <tr> <td width='100%' bgcolor='#ebecca'>
    <a href='$in_result->URL'target="_blank"><b>$in_result->title</b></a>
    </td></tr>
  <tr> <td> $in_result->snippet<br/> </td></tr>
  <tr> <td bgcolor='#fbfcda'>
    <a href='$in_result->URL'target="_blank">$in_result->URL</a></td>
  </tr></table><br/><br/>
EORESULT;
  }
 }
?>
```

利用上面的 google_keywords.inc.php 程序，制作一个 Web 网站搜索程序 searchGoogle.php，其页面含有一个输入搜索关键字的表单，提交搜索关键字后，调用 GoogleKeywords 类的 findAndPrintRelatedPages()方法，远程调用 Google 的 Web 服务，将搜索结果返回到本 PHP 程序，再传送到用户的浏览器上显示。程序内容见例 13.13。

【例 13.13】 搜索主程序（searchGoogle.php）。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>搜索结果</title>
</head>
<body>
<?php
// 显示此文的关键字
```

```
include('google_keywords.inc.php');
?>
<form name="form1" method="get" action="searchGoogle.php">
  输入关键字
  <input name="q" type="text" value="<? echo $_REQUEST["q"];?>" id="q">
  <input name="start" type="hidden" id="start" value="0">
  <input name="SBtn" type="submit" id="SBtn" value="搜索">
</form>
<?
// 显示相关的与关键字匹配的结果//
  if (isset($_REQUEST["q"])) {
    $q=$_REQUEST["q"];
    $start=$_REQUEST["start"];
    GoogleKeywords::findandPrintRelatedPages($q,$start);
  }
?>
<a href="searchGoogle.php?q=<? echo urlencode($q);?>&start=<? echo
$start-10;?>">上一页</a>
<a href="searchGoogle.php?q=<? echo
urlencode($q);?>&start=<? echo
$start+10;?>">下一页</a>
</body>
</html>
```

访问本地 Web 的 searchGoogle.php 程序，显示图 13.6 所示的表单界面，输入搜索关键字“Web 服务”后，单击“搜索”按钮，输出结果如图 13.7 所示。该搜索结果与直接访问 Google 网站得到的搜索结果相同。

图 13.6　searchGoogle.php 的访问界面

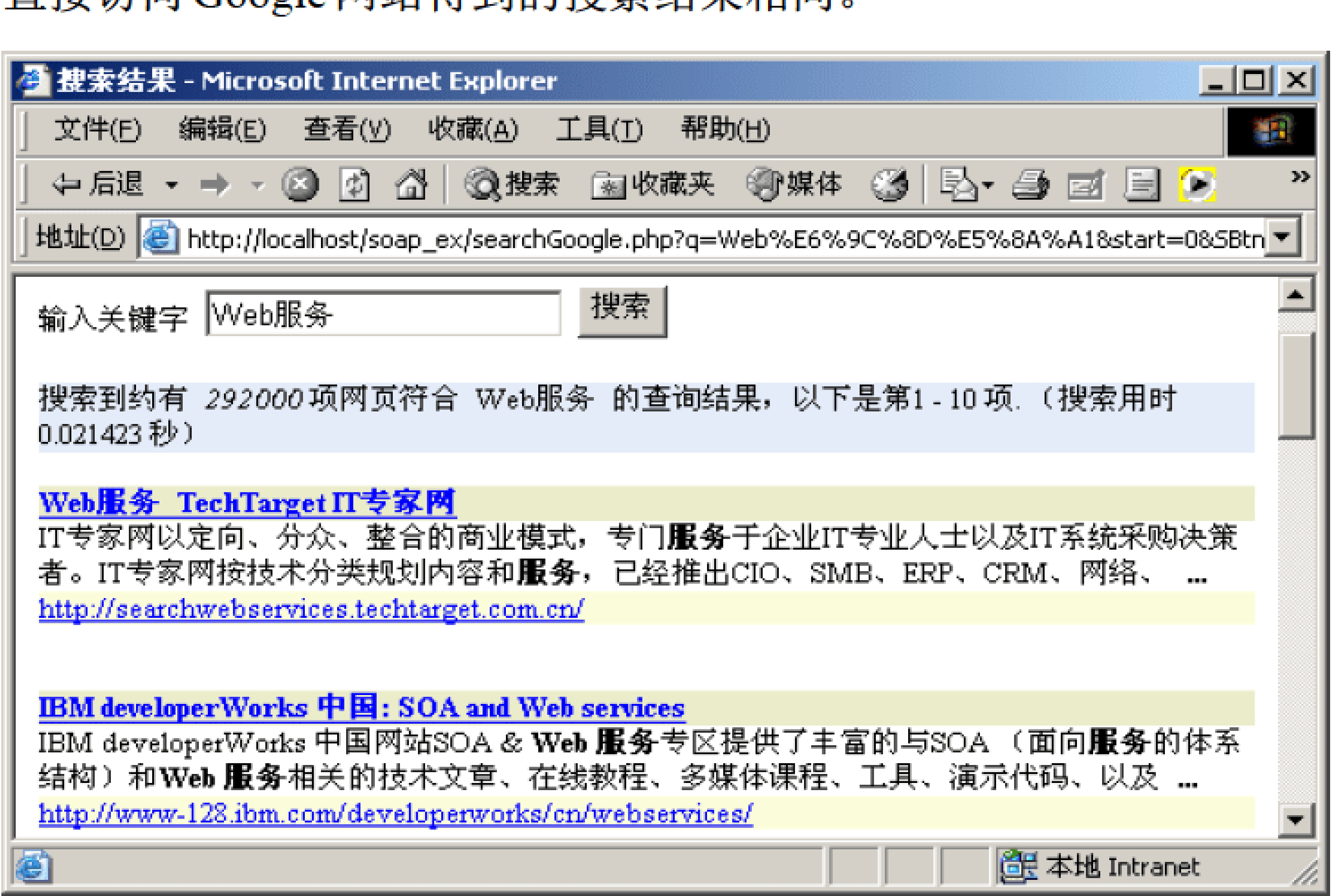


图 13.7　查找“Web 服务”的搜索结果

实　验　13

1．利用 PHP 5 和 XML-RPC 协议，编写两个 Web 服务，分别计算任意两个数的差、积。编写一个客户端程序，输入两个数，调用 Web 服务，得到两个数的差或积。

2．利用 PHP 5 和 SOAP 协议，编写一个 Web 服务，从 MySQL 的 student_db 数据库的 student 表中查询某一学生的学号、姓名、性别、出生日期。编写相应的 Web 服务客户端程序，输入学生的学号，调用 Web 服务，输出该学生的信息。

3．上机验证例 13.12 和例 13.13 程序。

习　题　13

1．什么是 Web 服务？

2．什么是远程过程调用？实现远程过程调用的常用两种通信协议是什么？

3．XML-RPC 和 SOAP 协议有何异同？

4．如何理解 XML-RPC 协议的工作原理？

5．如何理解 SOAP 协议的工作原理？

6．SOAP 协议由哪几个部分组成？

7．WSDL 协议的作用是什么？它由哪些部分组成？

第 14 章　PHP 和 Web 分布式数据交换

本章导读

Web 分布式数据交换（web distributed data exchange，WDDX）是一种以 XML 1.0 为基础，在 Web 应用程序和平台之间交换数据的技术，如 PHP、Java、ASP 和 Perl。WDDX 提供一种创建基于 XML 的数据结构的标准格式，能够在 Internet 上通过 HTTP、FTP、SMTP 和 POP 等协议来传输 WDDX 包。

PHP 的 WDDX 扩展提供了 PHP 数据和 WDDX 包之间互相转换的函数，将 PHP 数据转换为 WDDX 包的过程，称为串行化（serializing），将 WDDX 包转换为 PHP 数据类型的过程，称为反串行化（unserializing）。WDDX 函数使用平台无关的 WDDX 数据结构，支持 Web 应用程序之间的信息传输。

本章介绍 WDDX 的组成结构、WDDX 的数据类型，如何使用 PHP 5 的 WDDX 函数串行化数据和反串行化 WDDX 包以及如何利用 WDDX 创建 Web 服务。

14.1　WDDX 概述及其数据类型

14.1.1　WDDX 概述

WDDX，英文全称为 web distributed data exchange，是一种基于 XML 的 Web 分布式数据交换技术。1998 年，美国 Allaire 公司为了解决 ColdFusion 平台的分布式计算问题而建立了 WDDX。应用 WDDX，不同的 Web 系统之间可以方便地进行跨语言、跨系统、跨平台的复杂数据的交换与共享，对于提高 Web 站点的整体价值以及协助 Web 开发人员更加方便地构建 Web 系统有着重要的应用意义。

利用 WDDX，可以将一个 Web 应用程序的变量（包括变量名字、数据类型和值）串行化为 XML 文档（称为 WDDX 包），再发送到另一个 Web 应用程序。接收方的应用程序反串行化 XML 文档，重新构建为本地的变量和值。变量的数据类型可以是简单的数值、字符串类型，也可以是复杂的数据结构，如数组、结构和记录集。

WDDX 是平台和语言无关的，它允许各种平台的编程语言利用这一技术。因此，一个平台上以某一种语言编写的应用程序能够将数据传送到使用不同语言、不同平台的另一个应用程序。接收方的应用程序能够反串行化数据为本地的数据类型。

WDDX 不是正式标准，但它建立于开放标准，特别是以 XML 1.0 版为基础。WDDX 的发展和未来变化情况可追溯到开放项目——OpenWDDX.org (http://www.openwddx.org)。在该网站上可以找到更多的信息和软件开发工具包（SDK）。最近几年，几乎没有 WDDX 的研究活动。但并不意味着 WDDX 技术被放弃了，它仍然应用于许多平台和编程语言，特别是 PHP。

WDDX 基于 XML 1.0 版本，使用 HTTP 协议，在 Web 应用程序之间进行数据传输。利用串行化和反串行化函数，将本地数据串行化为 WDDX 包格式和将 WDDX 包格式的数据反串行化为本地数据类型的过程，如图 14.1 所示。

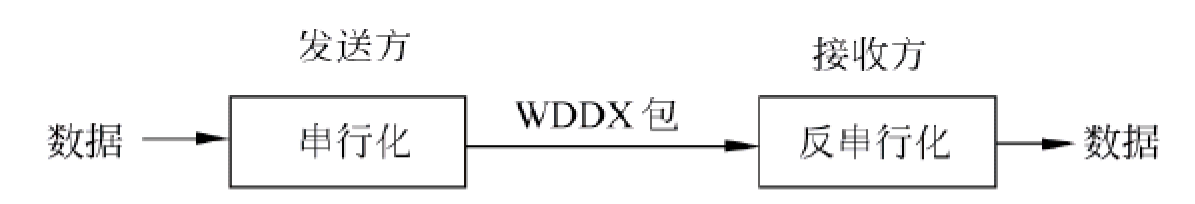


图 14.1 使用 WDDX 包交换数据的过程

串行化过程接收数据，将数据转换为 WDDX 包。WDDX 包的数据可以是简单数据类型或者复杂的数据类型。简单数据类型包括字符串、数值和布尔值。复杂数据类型是数组、结构和记录集。反串行化过程将 WDDX 包的数据转换为本地的原数据类型。

14.1.2 WDDX 数据类型

WDDX 以 XML 格式存储的数据组成 WDDX 包。在 XML 格式中，类似于 HTML，在元素的开始标签和结束标签之间封装了数据。WDDX 支持与 Web 编程语言（如 PHP）的数据类型相似的数据类型。WDDX 支持的数据类型如下所述。

- number：表示数值类型，包括整数和浮点数。其值范围是[–1.7E+308,–1.7E–308]和[1.7E–308,1.7E+308]。
- dateTime：以 ISO8601 编码格式表示的日期和时间值。对于月、日、小时、分钟、秒，其数字位为一位时，补上前缀 0。
- string：表示字符串类型。可以包含任意多个字符，字符串类型不含 null 值。可以把双字节字符编码为字符串。
- binary：表示二进制类型。其值为一串二进制数据，二进制数据编码为 base64 格式。
- array：表示数组类型。表示一组数据的集合，数组元素的下标为整数。
- struct：结构类型，表示一组任意类型的数据组成的对象集合。
- recordset：记录集，将数据表示为表格形式。记录集是一组行集合，每行由多个字段组成。在记录集中只能存储简单的数据类型，为了在表格形式中存储复杂的数据类型，应该使用结构数组。
- null：表示其值是未确定的。注意，null 值与数值 0 或空字符串是不同的。
- WDDX 提供的几种抽象数据类型，与其他编程语言的数据类型之间的对应关系如表 14.1 所示。

表 14.1 WDDX 数据类型和其他编程语言的数据类型的对应关系

WDDX 数据类型	PHP 数据类型	Java 数据类型
null	NULL	null
boolean	boolean	java.lang.Boolean
number	integer, float, double	java.lang.Double
dateTime		java.lang.Date
string	string	java.lang.String
array	array	java.lang.Vector
struct	array, object	java.lang.Hashtable
recordset		com.allaire.util. wddxRecordset
binary		com.allaire.util.Binary

14.2　WDDX 的结构

14.2.1　WDDX 结构导引

自从 1999 年公布 WDDX 1.0 以来，WDDX 文档的结构没有被改变过。WDDX 文档的复杂性依赖于被串行化的数据。变量的结构（如包括结构、多维数组和对象）越复杂，WDDX 文档的结构就越复杂。使用 WDDX 的数据交换通过 WDDX 包来实现。WDDX 包是一个 WDDX 格式的 XML 文档，它以 wddxPacket 元素为文档元素，文档元素包含一个 header 元素和一个 data 元素，header 元素是注释的容器，data 元素是要交换的实际数据的容器。wddxPacket 元素有一个 version 属性，指定 WDDX 的版本。因为目前只有一个版本，因此其值为 1.0。使用 comment 元素，在 WDDX 包中增加注释，它是 header 元素的可选的子元素。如果没有使用 comment 元素，则 header 元素只是一个空元素。例如，一个没有 comment 元素的 WDDX 包的结构如下：

```
<wddxPacket version='1.0'>
  <header/>
  <data>
    <!-- 此处为 WDDX 数据 -->
  </data>
</wddxPacket>
```

含有 comment 元素的 WDDX 结构如下：

```
<wddxPacket version='1.0'>
  <header>
    <comment>
    这是包的注释内容
    </comment>
  </header>
  <data>
    <!-- 此处为 WDDX 数据 -->
  </data>
</wddxPacket>
```

其中，data 元素包含要交换的数据，它是 wddxPacket 元素的子元素。这些交换的数据是要增加到 WDDX 包的数据，用以下数据类型元素来定义： null、boolean、number、dateTime、string、array、struct、recordset、binary。这些数据类型元素是 data 元素中的子元素。

14.2.2　简单的数据类型元素

简单数据类型元素定义简单的结构，其内容不能包括其他数据类型。这些元素包括 null、boolean、number、dateTime 和 string。

1．null 元素

null 元素表示一个 NULL 值，这依赖于所用的编程语言是否支持 NULL 类型。PHP 支

持 NULL，但要注意的是，如果和另一个使用不同编程语言的系统进行交换数据，当反串行化 NULL 时，null 可能被解析为一个空字符串，而不是 NULL。其语法为：

```
<null/>
```

2．boolean 元素

boolean 元素表示布尔值，其值可以是 true 或 false。WDDX 规定布尔值必须是小写的字母，才是有效的。该元素的语法为：

```
<boolean>false</boolean>
```

3．number 元素

number 元素表示浮点数字。在 PHP 中，它包括整数类型和浮点类型。其值范围是[–1.7E+308,–1.7E–308]和[1.7E–308,1.7E+308]。小数点后最多有 15 位小数。它与 8 字节的浮点数兼容，8B 的浮点数是 PHP 的浮点数的最大值。

下面的元素表示数字 12 345、–12.345 和 123 456 789 012 345 的串行化值。

```
<number>12345</number>
<number>-12.345</number>
<number>1.2345678901235E+014</number>
```

4．dateTime 元素

dateTime 元素以 ISO 8601 格式表示日期和时间。虽然 PHP 没有相应的 datetime 类型。但可以用 PHP 5 的 date('c')函数或者 date(DATE_ISO8601)来设置 ISO 8601 格式的日期和时间。该元素的语法为：

```
<dateTime>2005-10-08T17:28:04-04:00</dateTime>
```

5．string 元素

string 元素包含任意长度的字符串，字符串中不能含有 NULL。可以用 char 子元素来表示十六进制数为 00～1F 范围内的 UTF–8 格式的控制符。char 元素有一个 code 属性，该属性值是十六进制数表示的字符。因此，当设置一个含有特殊控制符的字符串时，string 元素的值可以包括普通字符和控制字符。下面的例子说明了如何使用 string 元素和 char 元素。

```
<string>这是一个没有任何控制字符的字符串</string>
<string>第 1 行<char code="0D"/><char code="0A"/>第 2 行</string>
```

14.2.3　复杂的数据类型元素

复杂的数据类型元素包括 array、struct、recordset 和 binary 元素。这些元素用于表示更加复杂的数据结构，如 PHP 的数组、类。只有 array 和 struct 这两个元素可以直接映射为本地 PHP 的类型，其余元素可以通过程序转换为应用程序可用的数据。

1．array 元素

array 元素保存数组元素下标为整数的数组的数据。在 PHP 中，数组元素的下标可以

是数字或者字符串。只有数字下标的数组才能映射为 array 元素，以字符串为下标的数组映射为 struct 元素。

说明：在 PHP 中，数组的数字下标从 0 开始编号。如果创建的数组，其下标不是从 0 开始，即使下标是数字，也不能映射为 array 元素。例如，使用 WDDX 扩展来串行化数组 array（2=>'a', 4=>'b', 6=>'c'）和 array（0=>'a', 2=>'b'），生成的是一个 struct 元素，而不是生成 array 元素。

array 元素的子元素表示数组元素值，子元素的类型可以是简单的数据类型和复杂的数据类型。也就是说，array 元素可以有一个或多个数据类型的子元素。array 元素有一个 length 属性，表示数组元素的个数。在 PHP 中，length 属性的值是在数组中调用 count()函数的值。例如，下面的两个 PHP 数组，其下标都是数字。

```
array('a', 1, false);
array(0=>'a', 1=>1, 2=>false);
```

串行化为 WDDX 包，生成的 array 元素都是如下结果。

```
<array length='3'>
<string>a</string>
<number>1</number>
<boolean value='false'/>
</array>
```

2．struct 元素

结构是以字符串为索引的数据集合。struct 元素表示结构化的内容。在 PHP 中，结构是以字符串为下标的关联数组和对象。要注意的是，数字下标从非零数字开始的数组，生成 struct 元素，而不是 array 元素。

struct 元素是 0 个或多个 var 元素的容器，这些元素表示变量或类的属性，其 name 属性存放变量名或类属性名。每个 var 元素的子元素是变量或类属性的值的元素，可以由任何数据类型元素组成。var 元素是 data 或 array 元素的子元素。例如，假设将以下 PHP 代码的变量串行化为一个 struct 元素。

```
$myint = 12345;
$mystring = "This is a string";
$mykeys = array('key1'=>1, 'key2'=>2);
```

那么，这些变量都被串行化为 var 元素，被包含在一个 struct 元素内，如下：

```
<struct>
<var name='myint'>
<number>12345</number>
</var>
<var name='mystring'>
<string>This is a string</string>
</var>
<var name='mykeys'>
```

```
<struct>
<var name='key1'>
<number>1</number>
</var>
<var name='key2'>
<number>2</number>
</var>
</struct>
</var>
</struct>
```

$mykeys 变量的串行化结果是一个复杂的数据结构，该变量的值是一个关联数组，因此，与$mykeys 变量对应的 var 元素不仅是 struct 元素的子元素，而且它也包含一个 struct 元素。该 struct 元素又包含另外的 var 元素，代表数组的每个数组元素。

由此可见，虽然 WDDX 结构的定义并不复杂，但是，对变量串行化后产生的 WDDX 文档可以是相当复杂的。串行化后的结构的复杂度直接与生成 WDDX 包的原始数据的复杂度有关。

3．recordset 元素

recordset 元素用于表示表格数据，它是二维的数据，如数据库的记录集。记录集是一组行集合，每行由多个字段组成。在记录集中只能存储简单的数据类型。

recordset 元素由任意多个 field 子元素组成，recordset 元素有两个属性：rowCount 属性定义行数，fieldNames 属性定义字段名列表，rowCount 属性的值是 recordset 元素内的数据行数，fieldNames 属性的值是用逗号分开的字段名列表。例如：

```
<recordset rowCount="2" fieldNames="ID,NAME,AGE">
<!-- field 元素内容 -->
</recordset>
```

根据这一 recordset 元素的属性，可知道该记录集包括两个记录，每个记录有三个字段，由 fieldNames 属性的字段名列表中指定，即 recordset 元素包含三个 field 元素。

field 元素包含记录集中指定字段的每行的值，它含有一个 name 属性，存放字段名，该字段名是 recordset 父元素的 fieldNames 属性值中的名字之一。field 元素的子元素是由任意多个简单类型的元素组成，即 null、boolean、dateTime、number、string 或 binary。每个 field 元素表示表格中某一列的各行数据。field 元素内的每个子元素只属于某一个数据类型。

例如，考虑以下来自数据库表的数据，如表 14.2 所示，该表有两行记录，每行分为为三个字段。

表 14.2　数据库表数据

ID	NAME	AGE
1	张小春	20
2	李大锋	25

在 WDDX 包中，当使用 recordset 元素存储该记录集时，数据被串行化为下列格式。

```
<recordset rowCount="2" fieldNames="ID, NAME,AGE">
<field name="ID">
<number>1</number>
<number>2</number>
</field>
<field name="NAME">
<string>张小春</string>
<string>李大锋</string>
</field>
<field name="AGE">
<number>20</number>
<number>25</number>
</field>
</recordset>
```

从这一 recordset 元素的结构可以看出，当逻辑读取这一文档时，必须对指定字段的每行数据进行处理，而不能直接地逐行地处理每行数据。正因为如此，用 struct 元素来存储表数据，会更加便于读取各行数据。

4．binary 元素

binary 元素表示二进制大对象（BLOB）数据，如图形数据，它是二进制数据表示的字符串。WDDX 1.0 要求对二进制数采用 Base-64 编码。binary 元素有三个属性：encoding 属性设置编码类型，在 WDDX 1.0 中，其值为 base64；length 属性指定二进制数据的长度；type 属性设置二进制数据的 MIME 类型。PHP 没有本地的 binary 类型，因此要用 PHP 字符串来表示 binary 数据。例如：

```
<binary encoding="base64" length="9312164" type="video/mpeg">
<!-- 此处为 Base64 编码格式的二进制数据 -->
</binary>
```

14.3 PHP WDDX 函数的使用

虽然可以用 PHP 5 的 XML 解析器来处理 WDDX 文档，但是利用 PHP 5 的 WDDX 扩展来处理 WDDX 文档更加快速、简单，能够完成串行化和反串行化过程。当使用这一 WDDX 扩展时，不能创建 WDDX 数据类型为 recordset 的元素，但是可以将 WDDX 包的数据反串行化为 PHP 的数据类型。

为了使用 PHP 5 的 WDDX 扩展功能，需要安装 WDDX 扩展模块。在 Windows 平台，默认时，PHP 内置了 WDDX 扩展，因此不需要修改 php.ini 文件。在 UNIX 和 Linux 等其他平台，必须在配置、编译 PHP 时，加上--enable-wddx 配置选项，以便 PHP 支持 WDDX。此外，WDDX 扩展是在 XML 扩展之上构建的，也必须安装 XML 扩展。

为了阅读方便，下面各个例子的程序运行结果是经过格式化输出的。因为 WDDX 串行化产生的 WDDX 文档为一串文本，不增加任何附加的空格、缩进和换行符。

14.3.1 用 WDDX 串行化数据

可以用两种方法将数据串行化为 WDDX 包。第 1 种方法是使用 wddx_serialize_value()、wddx_serialize_vars()函数，这类函数根据传递的参数建立一个完整的 WDDX 包。第 2 种方法是利用 wddx_packet_start()函数创建一个含有结构的 WDDX 包，然后多次调用 wddx_add_vars()函数，将多个变量添加到结构中。究竟使用哪种方法，依赖于应用程序处理数据的情况而定。

1. 简单串行化

简单串行化包括调用 wddx_serialize_value()或 wddx_serialize_vars()函数将数据串行化为完整的 WDDX 包。这两个函数的语法如下：

1）wddx_serialize_value()函数

格式：

```
string wddx_serialize_value(mixed var [, string comment])
```

该函数创建一个含有一个值的 WDDX 包。返回值是一个表示 WDDX 包的字符串。var 参数为被串行化的数据，可以是任何类型的数据。可选的 comment 参数指定包的注释，并添加到 WDDX 包的 header 元素内。

【例 14.1】 wddx_serialize_value()函数的示例（ex14_1.php）。

```
<?
$myArray = array(1,2,3);
print wddx_serialize_value($myArray, 'WDDX Packet');
?>
```

访问该程序，返回的 WDDX 包如下：

```
<wddxPacket version='1.0'>
<header>
<comment> WDDX Packet</comment>
</header>
<data>
<array length='3'>
<number>1</number>
<number>2</number>
<number>3</number>
</array>
</data>
</wddxPacket>
```

2）wddx_serialize_vars()函数

格式：

```
string wddx_serialize_vars(mixed var1 [, mixed var2, ...])
```

该函数创建一个含有结构的 WDDX 包，该结构中包括传递给函数的变量的串行化数据元素。返回值是一个 WDDX 包字符串。var1、var2 等是被串行化的多个变量名参数，每个参数

可以是变量名字符串，或者是一个包含变量名或另一个数组的字符串作为数组元素的数组。

一般情况下，使用 wddx_serialize_vars()函数，允许按名字串行化任意多个变量，创建一个 struct 元素，传递给该函数的每个变量创建为一个 var 元素，将变量名映射为 var 元素的 name 属性，这些 var 元素作为 struct 元素的子元素。使用该函数时，不能在 header 元素内创建注释。如果要在 WDDX 中包括注释，可以用 wddx_serialize_value()函数或者下面的第 2 种方法来创建 WDDX 包。

【例 14.2】 wddx_serialize_vars()函数的示例（ex14_2.php）。

```
<?php
$a = 121;
$b = "this is a string";
$c = array("blue", "orange", "green");
$d = "colors";
$vars = array("c", "d");
print wddx_serialize_vars("a", "b", $vars);
?>
```

程序生成的 WDDX 包含有一个结构，输出的 WDDX 包内容如下：

```
<wddxPacket version='1.0'>
<header/>
<data>
<struct>
<var name='a'><number>121</number></var>
<var name='b'><string>this is a string</string></var>
<var name='c'>
<array length='3'>
<string>blue</string><string>orange</string><string>green</string>
</array>
</var>
<var name='d'><string>colors</string></var>
</struct>
</data>
</wddxPacket>
```

【例 14.3】 串行化一个类对象（ex14_3.php）。

```
<?php
class myClass {
    public $prop1;
    public $prop2 = 'default';
    public $prop3 = 0;
}
$objMyClass = new myClass();
$myInteger = 2;
print  wddx_serialize_vars('myInteger', 'objMyClass');
?>
```

该程序串行化一个整数变量$myInteger 和一个对象$objMyClass。返回一个生成的 WDDX 包，如下：

```
<wddxPacket version='1.0'>
<header/>
<data>
<struct>
<var name='myInteger'><number>2</number></var>
<var name='objMyClass'>
<struct>
<var name='php_class_name'><string>myClass</string></var>
<var name='prop1'><null/></var>
<var name='prop2'><string>default</string></var>
<var name='prop3'><number>0</number></var>
</struct>
</var>
</struct>
</data>
</wddxPacket>
```

在输出结果中，$myInteger 变量和$objMyClass 变量被设置为最顶层 struct 元素之下的 var 子元素。$objMyClass 对象被串行化一个结构元素。可能感到奇怪的是 name 属性值为“php_class_name”的 var 元素。当使用 WDDX 扩展串行化一个对象时，name 属性值为“php_class_name”的 var 元素被添加到 struct 元素的第 1 个子元素。该元素为 string 类型，值为对象所在的类名。当反串行化 WDDX 包时，对象可以用返回的值进行实例化。

2．复杂的串行化

复杂串行化并不是说串行化数据很复杂，而是指在串行化时如何处理复杂的数据。以这种方式处理串行化，需要使用：wddx_packet_start()、wddx_add_vars()和 wddx_packet_end()三个函数。

1）wddx_packet_start()函数

格式：

```
int wddx_packet_start([string comment])
```

该函数创建一个新的 WDDX 包，并自动创建一个 WDDX 包的结构定义。返回值为一个整数，表示 WDDX 包标识符，此标识符用于 wddx_add_vars()和 wddx_packet_end()函数。可选参数 comment 表示注释字符串。

2）wddx_add_vars()函数

格式：

```
bool wddx_add_vars(int packet_id, mixed var1 [, mixed var2,...])
```

该函数串行化一个或多个变量，并添加到指定的 WDDX 包。返回值为 TRUE 或 FALSE。packet_id 参数是 wddx_packet_start()函数创建的 WDDX 包标识符，var1、var2 等是变量名字符串或者字符串数组，其用法与 wddx_serialize_vars()的变量相同。

3）wddx_packet_end()函数

格式：

```
string wddx_packet_end(int packet_id)
```

该函数关闭由 WDDX 包标识符指定的 WDDX 包，返回一个表示 WDDX 包的字符串。

为了创建一个 WDDX 包，用 wddx_add_vars()函数添加变量到 WDDX 包中，编程步骤如下：

（1）调用 wddx_packet_start()函数，创建一个空的 WDDX 包。

（2）添加数据到 WDDX 包。使用 wddx_add_vars()函数将每个变量添加到 WDDX 中。

（3）调用 wddx_packet_end()函数，关闭 WDDX 包。

【例 14.4】 用 wddx_add_vars()添加多个变量到 WDDX 包（ex14_4.php）。

```
<?php
$name="黄大山";
$department="计算机系";
$age="28";
$id=wddx_packet_start("个人信息");
wddx_add_vars($id, "name", "department", "age");
$packet=wddx_packet_end($id);
$fp=fopen("example14_4.xml", "w+");
fwrite($fp, $packet, strlen($packet));
fclose($fp);
print $packet;
?>
```

该程序串行化三个变量 name、department 和 age，将串行化的结果添加到<struct>元素的 var 子元素中。返回的 WDDX 包内容如下：

```
<wddxPacket version='1.0'>
<header>
<comment>个人信息</comment>
</header>
<data>
<struct>
<var name='name'><string>黄大山</string></var>
<var name='department'><string>计算机系</string></var>
<var name='age'><string>28</string></var>
</struct>
</data>
</wddxPacket>
```

【例 14.5】 查询 MySQL 的 student_db 数据库的 student 表的内容，生成 WDDX 包。

```
<?
$conn=mysql_connect("localhost","root","123456") or die('连接数据库服务器失
败');
mysql_select_db("student_db") or die('不能打开数据库');
mysql_query("SET NAMES 'gbk'");
$query="select Sno,Sname,specialty from student";
$result=mysql_query($query,$conn) or die('查询失败:'.mysql_error());
$row=0;
```

```
$wddxid = wddx_packet_start('学生信息');
$rowCount = mysql_num_rows($result);
$fieldCount = 3;
wddx_add_vars($wddxid, 'rowCount', 'fieldCount');
while ($line = mysql_fetch_array($result, MYSQL_ASSOC)) {
    $row++;
    $varname='row'.$row;
    $$varname=$line;    //使用变量的变量,将数组赋给$row1、$row2 等变量
    wddx_add_vars($wddxid,$varname);   //$row1、$row2 等变量添加到 WDDX 包
}
mysql_free_result($result);
$packet = wddx_packet_end($wddxid);
$fp=fopen("example14_5.xml", "w+");
fwrite($fp, $packet, strlen($packet));
print $packet;
?>
```

程序中，使用 mysql_fetch_array()函数从查询结果集读取一行数据，存放到关联数组$line，数组元素的下标为相应的字段名。在循环体中，使用变量的变量的方式（因 var 元素必须有唯一的 name 属性），一次将一个数组添加到 WDDX 包中。添加完数据后，调用 wddx_packet_end()函数关闭 WDDX 包。访问该程序，产生的 WDDX 包内容如下：

```
<wddxPacket version='1.0'>
<header>
<comment>学生信息</comment>
</header>
<data>
<struct>
<var name='rowCount'><number>6</number></var>
<var name='fieldCount'><number>3</number></var>
<var name='row1'>
<struct>
<var name='Sno'><string>0601</string></var>
<var name='Sname'><string>李大伟</string></var>
<var name='specialty'><string>会计</string></var>
</struct>
</var>
  ⋮
</data>
</wddxPacket>
```

14.3.2 反串行化数据

反串行化是将一个 WDDX 包转换为一个相关的 PHP 变量的过程。可以用 PHP WDDX 扩展的 wddx_deserialize()函数反串行化 WDDX 包，得到相应的本地 PHP 数据类型。其函

数格式如下：

```
mixed wddx_deserialize(string packet)
```

该函数将 WDDX 包反串化为 PHP 的本地数据。返回的值可以是字符串、数值或数组。WDDX 包的结构反串行化为关联数组。

说明： PHP 5.1 中，wddx_deserialize()函数是 wddx_unserialize()函数的别名。当运行于 PHP 5.1 以上版本时，可以使用 wddx_unserialize()函数。

【例 14.6】 将例 14.2 生成的 WDDX 包反串行化（ex14_6.php）。

```
<?php
$fp = fopen("http://localhost/PHP5BOOK/CH14/ex14_2.php", "r");
if ($fp) {
    $wddx = "";
    while(!feof($fp)) $wddx .= fread($fp, 4096);
    fclose($fp);
    $data=wddx_deserialize($wddx);
    var_dump($data);
    if(is_array($data)) {
        foreach ($data AS $key=>$value) {
            $$key = $value;
        }
        echo 'a='.$a."<br>";
        echo 'b='.$b."<br>";
        echo 'd='.$d."<br>";
        echo $c[0]."<br>";
        echo $c[1]."<br>";
        echo $c[2]."<br>";
    }
}
?>
```

因为 WDDX 包内容是一个结构元素，当反串行化 WDDX 包时，结构元素被转换为关联数组。执行 var_dump()函数，显示的结果如下：

```
array(4) {
  ["a"]=>
  int(121)
  ["b"]=>
  string(16) "this is a string"
  ["c"]=>
  array(3) {
    [0]=>
    string(4) "blue"
    [1]=>
    string(6) "orange"
```

```
      [2]=>
      string(5) "green"
    }
    ["d"]=>
    string(6) "colors"
  }
```

从这一结果可以看到，反串行化的结果是一个含有四个元素的数组，因为 WDDX 包内有 var 元素，产生的数组是一个关联数组，原来串行化时的变量名成为数组元素的下标，这使得重新构建原来的变量变得非常简单。

执行 foreach 循环语句，利用变量的变量这一方法，重新构建原来串行化时的变量，因此，产生$a、$b、$c 和$d 四个变量，其中$c 是一个一维数组。六个 echo 语句输出这四个变量的值与例 14.2 程序的变量值相同。

14.4　创建基于 WDDX 的 Web 服务

在这一节，通过一个实例介绍如何利用 PHP 5 和 WDDX 扩展模块来构建一个 Web 服务程序和对应的一个 Web 服务客户端程序，实现两个网站的 Web 应用程序之间的数据交换。

下面以一个 MySQL 数据库系统中创建的学生学籍管理 student_db 数据库的学生基本表 student 表为例，student 表含有 sno、sname、birthdate、specialty 和 native_place 等字段，分别存放学生的学号、姓名、出生日期、专业和籍贯内容。要求编写一个 Web 服务和一个 Web 服务的客户端程序。Web 服务能够根据客户端提供的学号 ID 值，在 student 表中查询某个学生的记录，然后将学生信息以 WDDX 包的格式发送给客户端。Web 服务的客户端提供待查询的学生学号 ID 值，将学号以 WDDX 包格式发送给 Web 服务，然后等待 Web 服务返回的 WDDX 包，再进一步处理、显示查询结果。

14.4.1　创建一个 WDDX Web 服务

Web 服务只接受客户端发来的 POST 请求，此请求包括了一个 WDDX 包，包中含有一个结构和一个 var 元素，var 元素的 name 属性设为 id，值为要查询的学号。Web 服务根据 id 的值，检索数据库中 student 表的 sno、sname、birthdate、specialty 和 native_place 等字段内容，将查询结果记录集作为一个二维的关联数组，串行化为 WDDX 包的若干个结构，最后输出 WDDX 包，以便客户端能够接收到此 WDDX 包。

【例 14.7】 一个简单的 WDDX Web 服务程序（wddxServer.php）。

```
<?php
//文件名: wddxServer.php
function getDBData($id) {
    if (is_numeric($id)) {
        $conn = mysql_connect('localhost', 'root', '123456') or die
        ("不能连接到 MySQL 数据库服务器");
        mysql_select_db("student_db",$conn);
```

```
            mysql_query("SET NAMES 'gbk'");
            $sql="select sno,sname,birthdate,specialty,native_place from
            student where sno='".$id."'";
            /* 从数据库检索数据,返回结果为串行化过的 WDDX 包。如果找不到记录,返回的 WDDX
            包是 NULL 值。*/
            $result = mysql_query($sql,$conn);
            $wddxid = wddx_packet_start("学生信息");
            while ($row = mysql_fetch_array($result, MYSQL_ASSOC)) {
                $row["sname"]=utf8_encode($row["sname"]);
                $row["specialty"]=utf8_encode($row["specialty"]);
                $row["native_place"]=utf8_encode($row["native_place"]);
                wddx_add_vars($wddxid,"row");
            }
            mysql_free_result($result);
            $packet = wddx_packet_end($wddxid);
            $fp=fopen("example_server.xml", "w");
            fwrite($fp, $packet, strlen($packet));
            fclose($fp);
            return $packet;
            //return wddx_serialize_value($row);
        } else {
            return wddx_serialize_value(NULL);
        }
}
/* 主程序: 只接受 POST 请求以及包变量的数据集*/
if (isset($_POST['packet'])) {
    $wddx_packet =$_POST['packet'];
    $wddx_packet = str_replace("\\", "", $wddx_packet);//删除 WDDX 包中的反
    斜线
    $arrdata = wddx_deserialize($wddx_packet);
    /* 根据请求 id,检索数据表,返回生成的 WDDX 包*/
    if ($wddx_packet && $arrdata) {
        if (is_array($arrdata) && array_key_exists('id', $arrdata)) {
            print getDBData($arrdata['id']);
            exit;
        }
    }
} else {
/* 如果请求有错,发送一个值为 NULL 的包 */
    print wddx_serialize_value(NULL);
}
?>
```

程序说明：

（1）因为数据库存放的数据含有中文，因此利用 MySQL 的 SET NAMES 'gbk'命令，

将字符集设置中文 GBK 字符集。

（2）while 循环语句将查询结果集的每一行记录作为一个结构，添加到 WDDX 包中，因此 WDDX 可能含有多个结构。

（3）从客户端接收的 POST 请求内容（即 WDDX 包）中含有多余的反斜线，为了获取正确的 WDDX 包，利用 str_replace("\\", "", $wddx_packet)函数，删除 POST 请求中的全部反斜线（\）符号。

（4）查询结果中的 sname、specialty 和 native_place 字段内容都是中文，需要用 utf8_encode()将非 UTF-8 编码的中文文字转换为 UTF-8 编码的字符。这样，与客户端程序交换 WDDX 包，客户端才能识别中文。

（5）最重要的注意事项是用 Macromedia Dreamweaver MX 软件编辑该程序时，页面的字符集必须设置为 UTF-8。还需注意的是，程序中只能包含 PHP 程序代码，不能含有任何其他内容的代码（如 HTML 标记）。

14.4.2 编写 WDDX Web 服务的客户端程序

Web 服务的客户端程序很简单，它通过 Socket 与另一个 Web 服务器（其实也可以是同一个 Web 服务器）的 Web 服务建立连接，以 POST 请求方式向 Web 服务发送一个含有要查询的学生学号的 WDDX 包，然后等待接收 Web 服务返回的 WDDX 包。收到 WDDX 包后输出查询的结果。这一程序代码可以嵌入到 HTML 网页中，参见例 14.8 的代码内容。

【例 14.8】 WDDX Web 服务的客户端程序（wddxClient.php）。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>WDDX Web 服务客户端</title>
</head>
<body>
<?php
/* 远程服务器地址,指定远程服务程序所在的服务器名和端口号*/
$remote_protocol = 'tcp';
$remote_server = 'localhost';
$remote_server_port = 80;
/* 串行化包.本例中,请求查询记录的 ID 值为 0603 */
$packet = wddx_serialize_value(array('id'=>'0603'));
/* 通过 Socket 与远程服务器建立连接 */
$remote_connect = $remote_protocol.'://'.$remote_server;
$sock = fsockopen($remote_connect, $remote_server_port, $errno, $errstr,
30) or die("不能连接到 WDDX 服务端");
if (!$sock) die("$errstr ($errno)\n");
/* 发送含有 WDDX 包的 POST 请求 */
$data = 'packet='.$packet;
fwrite($sock, "POST /PHP5BOOK/CH14/wddxServer.php HTTP/1.0\r\n");
fwrite($sock, "Host: $remote_server\r\n");
```

```
fwrite($sock, "Content-type: application/x-www-form-urlencoded\r\n");
fwrite($sock, "Content-length: " . strlen($data) . "\r\n");
fwrite($sock, "Accept: */*\r\n");
fwrite($sock, "\r\n");
fwrite($sock, "$data\r\n");
fwrite($sock, "\r\n");
/* 接收 Web 服务返回的报头和 WDDX 包格式的应答内容 */
$headers = "";
while ($str = trim(fgets($sock, 4096)))
    $headers .= "$str";
$packet = "";
while (!feof($sock))
    $packet .= fgets($sock, 4096);
fclose($sock);
/* 反串行化包数据,输出生成的数据 */
$arrData = wddx_deserialize($packet);
/*输出 Web 服务返回的记录集 */
if (is_array($arrData)) {
    if (count($arrData) > 0) {
        foreach ($arrData AS $rownum=>$arrRow) {
            foreach ($arrRow AS $fieldname=>$fieldvalue) {
                print $fieldname.": ".$fieldvalue."<br>\n";
            }
            print "\n";
        }
    } else {
        print "没有返回的记录";
    }
} else {
/* 如果出现错误,输出一些信息 */
    var_dump($arData);
}
?>
</body>
</html>
```

由于 Web 服务返回的 WDDX 包可能含有多个结构，因此，在客户端程序中，反串行化 WDDX 包得到的是二维数组，因而利用双重循环 foreach 语句输出这一个二维数组的各个元素值。

为了测试例 14.7 的 wddxServer.php 和例 14.8 的 wddxClient.php 这两个程序，要求将它们保存到本地网站的目录中，其中，wddxServer.php 文件存放本地网站根目录下的 /PHP5BOOK/CH14 子目录。如果存放到其他目录，需要对例 14.7 的相关内容作修改。

这一客户端程序是查询学号为“0603”的学生信息，访问该程序，输出结果如图 14.2 所示。

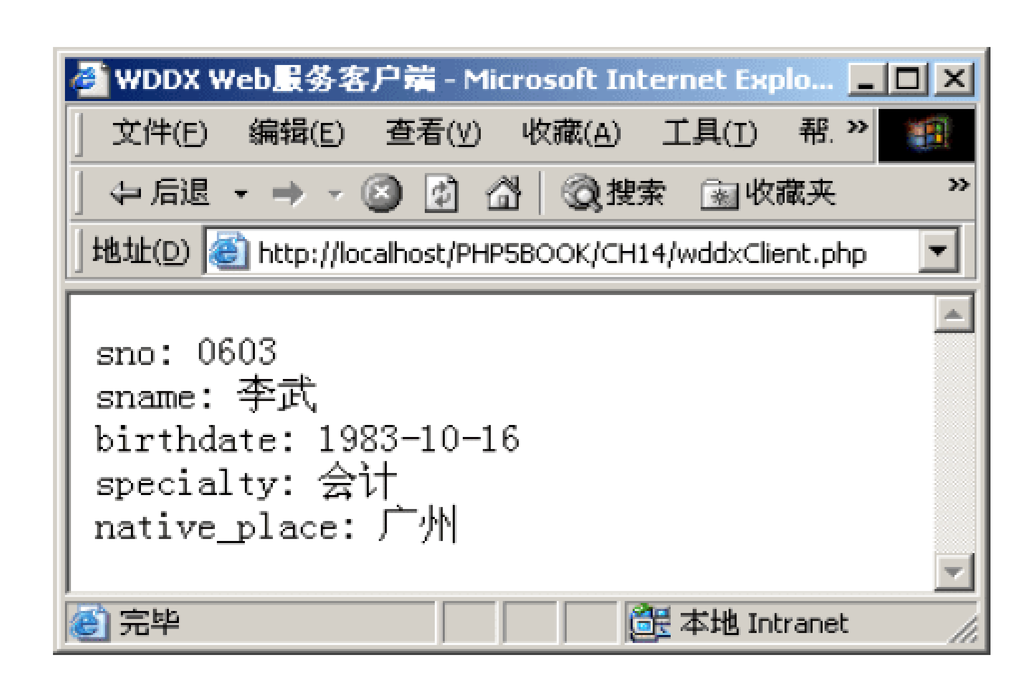


图 14.2　访问 wddxClient.php 的结果

对该程序作适当修改，增加表单，可以查询任何学生的信息。这一问题留给读者练习。

实　验　14

1．修改例 14.8 的程序，以便能够查询任何学生的信息。

2．针对 student_db 数据库的 student 表、course 表和 score 表的记录，编写一个 Web 服务程序，查询某一个学生的所有参加考试的课程成绩，查询结果包括学号、姓名、课程名和成绩字段的内容。再编写一个调用 Web 服务的客户端程序，输入待查询的学号，调用 Web 服务，得到查询结果。要求 Web 服务与客户端之间通过 WDDX 进行数据交换。

3．编写一个程序，实现：读取一个图形文件内容，将其串行化为 WDDX 包，然后显示反串行化后得到的图形。

习　题　14

1．什么是 WDDX？

2．如何理解 WDDX 的数据交换方式？

3．WDDX 规范包括哪些数据类型？

4．wddx_serialize_vars()和 wddx_serialize_value()函数有什么区别？

5．如何根据反串行化得到的字符串，恢复原来的变量？

第 15 章　XML 和数据库之间的数据交换

本章导读

在 Internet 的应用中，往往使用数据库来存储大量的数据，这些数据库运行于不同的平台。为了能够让不同平台的异构数据库之间能够互相交换数据，完成一个数据库的数据迁移到另一个数据库中，可以采用 XML 作为数据交换的信息载体，把一个数据库的数据转换为 XML 文档，然后将 XML 文档通过网络传输到另一个系统，再由另一系统平台的程序将 XML 文档导入到数据库。因此，可以通过中间件实现两个数据库之间的数据交换。

PHP 作为 Internet 上的编程语言，能够将存储在数据库的数据转换为 XML 文档，反之亦然。本章将介绍如何利用 PHP 的 SAX、DOM 和 SimpleXML 解析器，实现将 XML 文档存储到一个数据库中以及如何将数据库的数据导出为 XML 文档，同时也介绍如何将 XML 文档转换为 HTML 格式。

15.1　导出数据库数据到 XML 文档

15.1.1　导出 MySQL 数据库数据到 XML 文档

利用 PHP，可以导出和转换数据库的数据，存储为一个 XML 文档。也可以用各种工具，如 xpath 和 XSLT，对存储在数据库的数据进行操纵。从数据库导出数据到 XML 文档的编程过程为：

（1）建立与数据库的连接；

（2）通过执行 SQL 查询语句，从数据库检索数据；

（3）根据系统内存中的检索信息，建立一个 XML 文档。

可以使用 DOM、SAX 或 SimpleXML 来操作和解析生成的 XML 文档。

下面通过一个例子说明。首先在 MySQL 中建立一个 mydb 数据库，然后在 mydb 数据库中建立两个表：dept 表和 employee 表。dept 表用来存储部门的编号和部门名称，employee 表用来存储职工的职工编号、姓名、部门编号和工资。其 SQL 操作命令如下：

```
CREATE DATABASE mydb;
CREATE TABLE dept (
   deptID varchar(3) NOT NULL default NULL,
   deptName varchar(30) default NULL
 ) ENGINE=MyISAM DEFAULT CHARSET=utf8;
SET CHARACTER SET gb2312;
```

```
CREATE TABLE employee (
   EmpID varchar(5) NOT NULL default '',
   EmpName varchar(10) NOT NULL default '',
   deptID varchar(3) default NULL,
   Salary decimal(8,2) default NULL
  ) ENGINE=MyISAM DEFAULT CHARSET=utf8;

INSERT INTO employee VALUES ('1001', '张伟晓', '001', 1700.50);
INSERT INTO employee VALUES ('1002', '李小英', '001', 1900.00);
INSERT INTO employee VALUES ('1003', '黄佳佳', '002', 1600.50);
INSERT INTO dept VALUES ('001', '行政科');
INSERT INTO dept VALUES ('002', '生产科');
```

执行上述 SQL 命令，在 employee 表插入三个记录，在 dept 表插入两个记录，如表 15.1 和表 15.2 所示。

表 15.1　employee 表

EmpID	EmpName	deptID	Salary
1001	张伟晓	001	1700.50
1002	李小英	001	1900.00
1003	黄佳佳	002	1600.50

表 15.2　dept 表

deptID	deptName	deptID	deptName
001	行政科	002	生产科

说明：上述 CREATE 命令将表的默认字符集设置为 UTF8，其目的是利用 DOM、SAX 和 SimpleXML 解析器来解析数据表的记录时，能够正确地处理中文字符。

利用 DOM 解析器，可以将数据库表中的数据导出为 XML 树。为了完成这一功能，需要先建立与数据库的连接，然后用 PHP 的 MySQL 函数查询记录，根据查询的记录集建立 XML 文档。例 15.1 给出了如何从 MySQL 数据库中检索数据，建立 XML 文档的程序。

【例 15.1】 利用 DOM 解析器，根据数据库的 Employee 表数据，建立 XML 文档（ex15_1.php）。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>从 employee 表导出数据到 XML 文档</title>
</head>
<body>
<?php
$conn = mysql_connect("localhost", "root", "123456") or die ("不能连接到
MySQLeb 服务器!");
mysql_select_db("mydb") or die ("不能打开数据库!");
$query = "SELECT * FROM employee";
```

```
mysql_query("SET NAMES 'gbk'");
$result = mysql_query($query) or die ("查询错误: $query." .mysql_error());
if (mysql_num_rows($result) > 0)
{
   $dom = new DOMDocument('1.0',"utf-8");
   $dom->formatOutput = TRUE;
   $root = $dom->createElement('EmployeeInfo');
   $dom->appendChild($root);
      while(list($EmpID,$EmpName,$deptID,$Salary)=mysql_fetch_row($result))
   {
      $rec = $dom->CreateElement("Employee");     // 建立子节点
      $rec->setAttribute("EmpID", $EmpID);
      $rec->appendChild($dom->createElement("EmpName", $EmpName));
      $rec->appendChild($dom->createElement("deptID", $deptID));
      $rec->appendChild($dom->createElement("Salary", $Salary));
      $root->appendChild($rec);
   }
   $xml=$dom->saveXML();
   $dom->Save("employees.xml");
   echo $xml;   // 输出 XML 树
}
?>
</body>
</html>
```

该程序查询 mydb 数据库的 Employee 表，获取查询的数据，再建立 XML 文档。调用 DOMDocument 类构造方法创建一个 XML 树对象，增加根元素 EmployeeInfo。while 循环语句将查询结果集中的每个记录作为一个 Employee 子元素增加到 EmployeeInfo 元素。最后调用 saveXML()方法将 XML 树对象转换为字符串，以便用 echo 命令输出该字符串。访问该程序，输出的 XML 文档如下，在浏览器上显示的结果如图 15.1 所示。

```
<?xml version="1.0" encoding="utf-8"?>
<EmployeeInfo>
  <Employee EmpID="1001">
    <EmpName>张伟晓</EmpName>
    <deptID>001</deptID>
    <Salary>1700.50</Salary>
  </Employee>
  <Employee EmpID="1002">
    <EmpName>李小英</EmpName>
    <deptID>001</deptID>
    <Salary>1900.00</Salary>
  </Employee>
  <Employee EmpID="1003">
    <EmpName>黄佳佳</EmpName>
```

```
    <deptID>002</deptID>
    <Salary>1600.50</Salary>
  </Employee>
</EmployeeInfo>
```

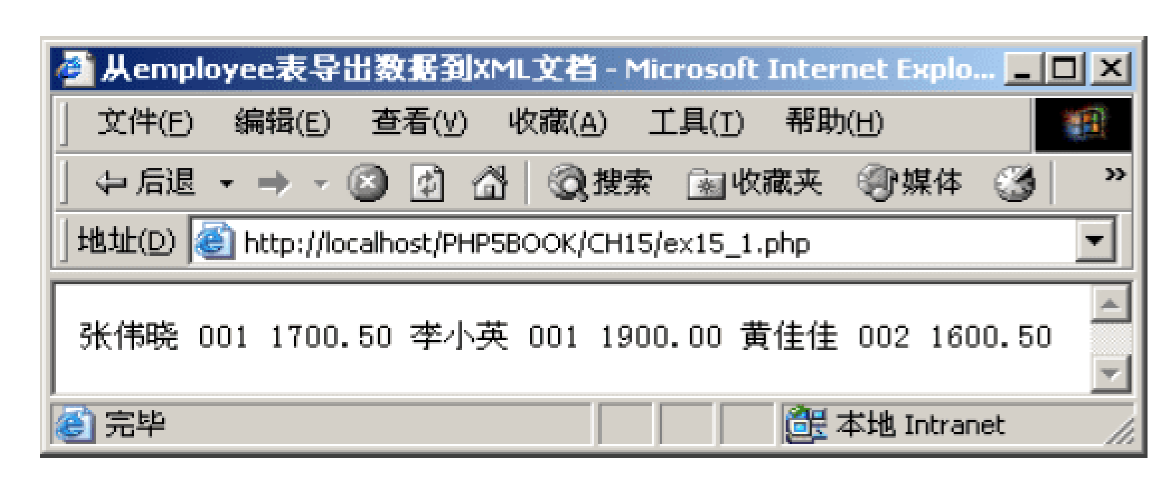


图 15.1　例 15.1 的运行结果

也可以同时从多个表导出数据。在 PHP 中，执行多表查询操作，得到检索的数据，根据这些数据来建立 XML 文档。例如，要显示职工的姓名和所在的部门名称。

【例 15.2】 从 employee 表和 dept 表导出数据，生成 XML 文档（ex15_2.php）。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>导出多表数据到 XML 文档</title>
</head>
<body>
<?php
$conn=mysql_connect("localhost","root","123456") or die("不能连接 MySQL!");
mysql_select_db("mydb") or die ("不能打开数据库!");
mysql_query("set names 'gbk'");
$query = "SELECT * FROM employee";
$result = mysql_query($query) or die ("查询错误: $query.".mysql_error());
if (mysql_num_rows($result) > 0)
{
   $dom = new DOMDocument('1.0',"utf-8");
   $dom->formatOutput = TRUE;
   $root = $dom->createElement('EmployeeInfo');
   $dom->appendChild($root);
   //  循环结果集
while(list($EmpID,$EmpName,$deptID,$Salary) = mysql_fetch_row($result))
   {
      $rec = $dom->CreateElement("Employee");      // 建立子节点
      $rec->setAttribute("EmpID", $EmpID);
      $rec->appendChild($dom->createElement("EmpName", $EmpName));
      $rec->appendChild($dom->createElement("Salary", $Salary));
      // 从 dept 表增加节点
      $query2 = "SELECT Deptname FROM dept where deptID = '$deptID'";
      $result2 = mysql_query($query2) or die ("Error in query: $query2. " .
                   mysql_error());
```

```
        while(list($Deptname) = mysql_fetch_row($result2))
          $rec->appendChild($dom->createElement("Deptname","$Deptname"));
        $root->appendChild($rec);
      }
  }
mysql_close($conn);
$xml=$dom->saveXML();
$dom->Save("ex5_2.xml");
echo $xml;   // 输出 XML 树
?>
</body>
</html>
```

访问该程序，输出 XML 文档如下，在浏览器上显示的结果如图 15.2 所示。

```
<?xml version="1.0" encoding="utf-8"?>
<EmployeeInfo>
  <Employee EmpID="1001">
    <EmpName>张伟晓</EmpName>
    <Salary>1700.50</Salary>
    <Deptname>行政科</Deptname>
  </Employee>
  <Employee EmpID="1002">
    <EmpName>李小英</EmpName>
    <Salary>1900.00</Salary>
    <Deptname>行政科</Deptname>
  </Employee>
  <Employee EmpID="1003">
    <EmpName>黄佳佳</EmpName>
    <Salary>1600.50</Salary>
    <Deptname>生产科</Deptname>
  </Employee>
</EmployeeInfo>
```

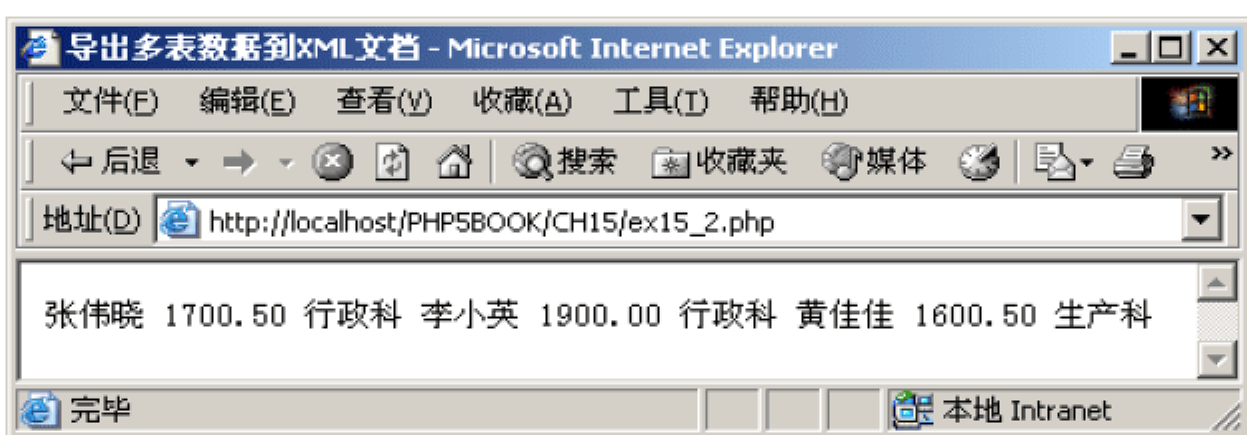


图 15.2 例 15.2 的运行结果

15.1.2 将 XML 文档转换为 HTML 格式

除了将数据库的数据导出为 XML 文档外，还可以将 XML 文档转换为 HTML 格式。

为了将生成的 XML 文档格式化为 HTML 格式，需要使用 SAX 解析器，编程步骤为：

（1）从数据库检索记录，生成 XML 文档；

（2）将生成的 XML 文档转换为 HTML 网页。

可以利用 SAX 解析器将 XML 文档转换为 HTML 格式，SAX 是一个基于事件的以块形式解析 XML 文档的一种方法，当解析器遇到 XML 文档的标记时，就调用回调函数，处理这些标记。

【例 15.3】 将 Employee 表的数据导出为 XML 文档，然后将 XML 文档转换为 HTML 格式（ex15_3.php）。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>将 XML 转换为 HTML</title>
</head>
<body>
<?php
// 定义数据库参数
$hostname = "localhost";
$username = "root";
$password = "123456";
$database = "mydb";
$table = "employee";
//  导出数据库记录,生成 XML 文档
$conn = mysql_connect($hostname, $username, $password) or die("不能连接
MySQL!");
mysql_select_db($database) or die ("不能打开 MySQL 数据库!");
mysql_query("SET NAMES 'gbk'");
$query = "SELECT * FROM $table";
$result = mysql_query($query) or die ("查询错误: $query. " . mysql_error());
if(mysql_num_rows($result) > 0)
{
   $dom = new DOMDocument('1.0',"utf-8");
   $dom->formatOutput = TRUE;
   $root = $dom->createElement("table");
   $root->setAttribute("name", $table);
   $dom->appendChild($root);
   $struct = $dom->createElement("struct");   // 建立节点
   // 为每个字段名、类型、宽度而建立元素,并添加到 XML 文档的 struct 元素
   $fields = mysql_list_fields($database, $table, $conn);
   for ($t=0; $t<mysql_num_fields($fields); $t++)
   {
      $field = $dom->createElement("field");  //一个字段的定义
      $name = mysql_field_name($fields, $t);
      $length = mysql_field_len($fields, $t);
```

```
            $type = mysql_field_type($fields, $t);
            $field->appendChild($dom->createElement("name", $name));
            $field->appendChild($dom->createElement("type", $type));
            $field->appendChild($dom->createElement("length", $length));
            $struct->appendChild($field);  //增加一个 field 元素到 struct 元素
        }
        $root->appendChild($struct);  //增加一个 struct 元素到 table 元素中
        $data = $dom->createElement("data");
        // 获取结果集每个记录,作为一个 record 元素,增加到 data 元素
        while($row = mysql_fetch_array($result,MYSQL_ASSOC))
        {
            $record =$dom->createElement("record");
            foreach ($row as $fieldname=>$fieldvalue)
                $record->appendChild($dom->createElement("item", $fieldvalue));
            $data->appendChild($record);
        }
        $root->appendChild($data);
        $xml_string = $dom->saveXML(); //将 XML 树转换为字符串
        echo $xml_string;
    }
    mysql_close($conn);
    // 利用 SAX 解析器将 XML 文档转换为 HTML 网页,以下数组保存开始标签的 HTML 标记
    $startTags = array(
            'TABLE' => '<html><head></head><body><table border="1"
            cellspacing="0"cellpadding="5">',
            'STRUCTURE' => '<tr>',
            'FIELD' => '<td bgcolor="silver"><font face="Times Roman" size="-1">',
            'RECORD' => '<tr>',
            'ITEM' => '<td><font face="Arial" size="-1">',
            'NAME' => '<b>',
            'TYPE' => '<i>(',
            'LENGTH' => ', '
    );
    //以下数组存放结束标签的 HTML 标记
    $endTags = array(
            'TABLE' => '</body></html></table>',
            'STRUCTURE' => '</tr>',
            'FIELD' => '</font></td>',
            'RECORD' => '</tr>',
            'ITEM' => ' </font></td>',
            'NAME' => '</b>',
            'TYPE' => '',
            'LENGTH' => ')</i>'
    );
    // 当找到开始标签时调用本函数
```

```
function startElementHandler($parser, $name, $attributes)
{
   global $startTags;
   if($startTags [$name])
   {
      //查找本开始标记对应的数组元素,输出相应的 HTML 标签
      echo $startTags[$name];
   }
}
// 当找到结束标签时调用本函数
function endElementHandler($parser, $name)
 {
   global $endTags;
   if($endTags [$name])
   {
      // 查找本结束标记对应的数组元素,输出相应的 HTML 标签
      echo $endTags [$name];
   }
}
// 当找到字符数据时调用本函数
function characterDataHandler($parser, $data)
{
   echo utf8_decode($data);
}
// 主程序: 初始化解析器
 $xml_string=utf8_encode($xml_string);
$xml_parser = xml_parser_create("utf-8");
// 设置回调函数
xml_set_element_handler($xml_parser, "startElementHandler",
"endElementHandler");
xml_set_character_data_handler($xml_parser, "characterDataHandler");
// 解析 XML 文档
if (!xml_parse($xml_parser, $xml_string))
{
   $ec = xml_get_error_code($xml_parser);
   die("XML 解析错误 (错误代码 " . $ec . "): " . xml_error_string($ec) .
   "<br>错误发生的行" . xml_get_current_line_number($xml_parser) . ", column "
   .xml_get_current_column_number($xml_parser) . ", byte offset " .
   xml_get_current_byte_index($xml_parser));
}
xml_parser_free($xml_parser);
?>
</body>
</html>
```

上述程序首先利用DOM解析器将Employee表的数据转换为XML文档，然后使用SAX解析器将生成的XML文档转换为HTML格式。图15.3是该程序的运行结果，其中前两行数据是输出XML文档的结果，后面的表格是XML文档转换为HTML表格形式。

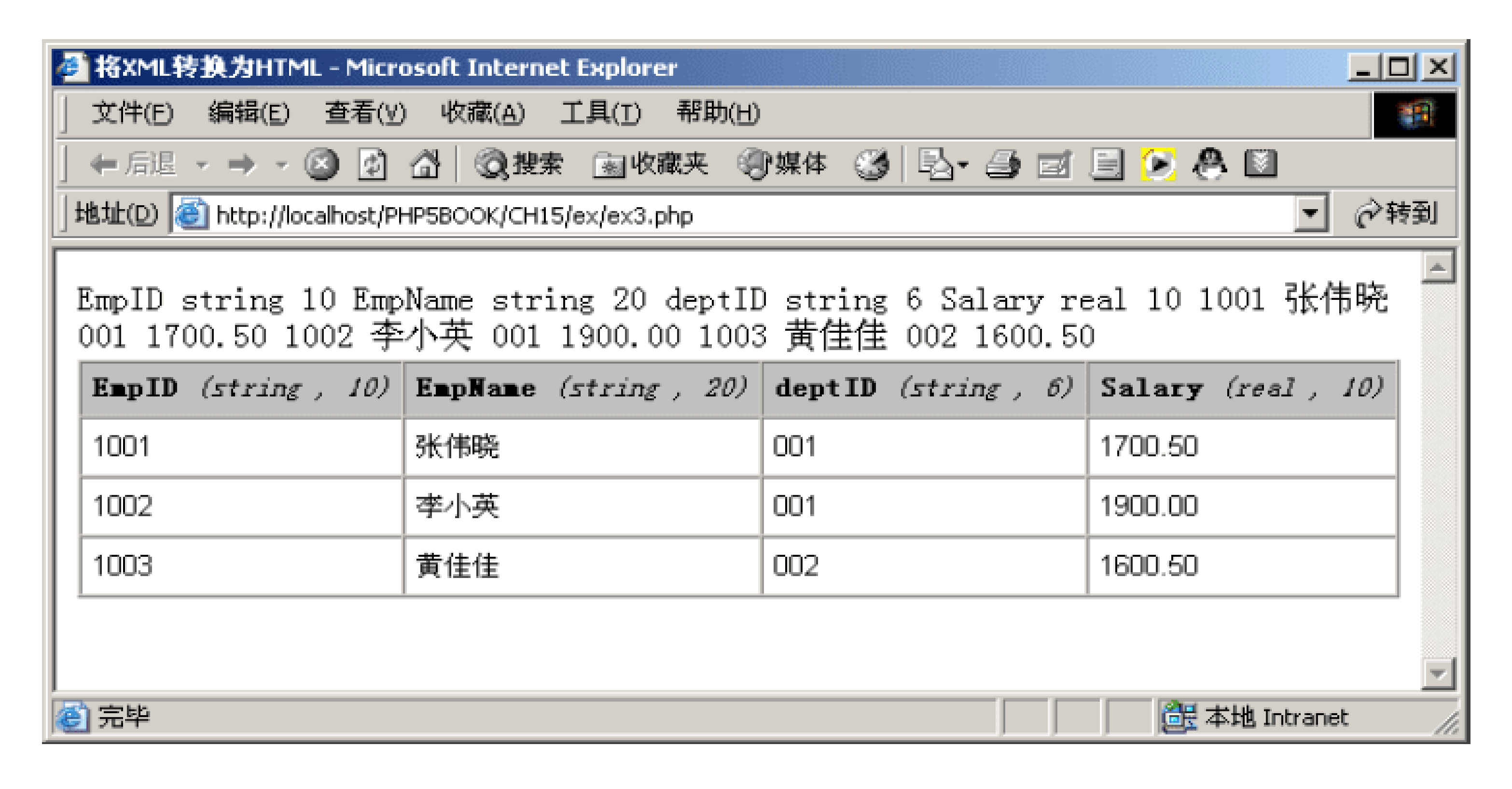


图 15.3　例 15.3 的运行结果

15.2　导入 XML 数据到 MySQL 数据库

15.2.1　用 SAX 导入数据到数据库

除了将数据库记录导出为 XML 文档之外，PHP 还可以导入 XML 文档到数据库中。利用DOM和SAX解析器，可以根据XML文档构造SQL的INSERT命令，并执行INSERT命令，将记录插入到数据库中。为了将 XML 文档导入到数据库中，可以按以下步骤编程：

（1）用 DOM 或者 SAX 解析 XML 数据；

（2）根据 XML 文档中存储记录的各字段的元素值，构造 INSERT 命令的字段列表和值列表；

（3）执行 MySQL 的 INSERT 命令；

（4）重复（2）、（3）两步，直到读完 XML 文档内容为止，最后关闭数据库连接。

例如，有一个 XML 文件 student.xml，该文件存放了学生的信息，包括姓名、年龄和地址，student.xml 文件的内容如下。

```
<?xml version="1.0" encoding="gb2312" standalone='yes'?>
<table name="student">
<record>
    <name>王超</name>
    <age>25</age>
    <address>广州市东风路</address>
</record>
<record>
    <name>徐小帆</name>
```

```
        <age>28</age>
        <address>广州市天河</address>
    </record>
    </table>
```

上述 XML 文档中，<table> 元素的 name 属性指定了表名，一个<record>元素指定一个学生信息（如 name、age、address）。

为了将 student.xml 文件的数据导入到 MySQL 的 mydb 数据库的 student 表，需要预先在 mydb 数据库中创建 student 表的结构，命令如下：

```
CREATE TABLE student (
    name Varchar(30) NOT NULL,
    age Integer NOT NULL,
    address Varchar(30) NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

上述命令建立一个 student 表，在其中，插入来自于 student.xml 文件的数据。因为 XML 文档中含有中文字符，因此，必须将该表的默认字符集设置为 utf8 编码，这样才能正确存储汉字。下面的例 15.4 给出了不需要指定表名，导入 XML 文档到数据库的程序。

【例 15.4】 用 SAX 解析 student.xml 文件，导入数据到 mydb 数据库的 student 表，程序如下（ex15_4.php）。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>用 SAX 导入 XML 文档到 MySQL 数据库</title>
</head>
<body>
<?php
// 初始化一些变量
$table="";
$cTag = "";
$fields = array();
$values = array();
$xml_file="student.xml";    // 指定要解析的 XML 文件
$host = "localhost";        // 数据库连接参数
$user = "root";
$password = "123456";
$db = "mydb";
// 当找到元素的开始标记时,调用本函数
function startElementHandler($parser, $name1, $attributes)
{
   global $cTag, $table;
   $cTag = $name1;
   if (strtolower($cTag)=="table")   // 获取表名
```

```
        {
          foreach ($attributes as $v)  $table=$v;
        }
    }
    // 当找到元素的结束标记时,调用本函数
    function endElementHandler($parser, $name1)
    {
      global $fields, $values, $count, $cTag;
      global $conn, $table;   // 导入数据库连接、表名
      if (strtolower($name1) == "record")
       {
        // 生成 INSERT 命令字符串
        $query = "INSERT INTO $table";
        $query .= "(" . join(", ", $fields) . ")";
        $query .= " VALUES(\"" . join("\", \"", $values) . "\");";
        // 执行查询
        mysql_query($query) or die ("查询错误: $query. " .mysql_error());
        // 复位内部计数器数和数组
        $fields = array();
        $values = array();
        $count = 0;
        $cTag = "";
       }
     }
    // 当找到元素的字符数据时调用本函数
    function characterDataHandler($parser, $data)
    {
      global $fields, $values, $cTag, $count;
      if (trim($data) != "")
      {
        // 增加字段名-字段值对到$fields 和$values 数组
        $fields[$count] = $cTag;
        //用转义字符表示特殊字符
        $values[$count] = mysql_escape_string($data);
        $count++;
      }
    }
    // 初始化 SAX 解析器
    $xml_parser = xml_parser_create();
    // 设置回调函数
    xml_set_element_handler($xml_parser, "startElementHandler",
    "endElementHandler");
    xml_set_character_data_handler($xml_parser, "characterDataHandler");
    $conn=mysql_connect($host,$user,$password) or die("不能连接 MySQL");
    mysql_query("set names 'utf8'");
```

```
mysql_select_db($db) or die ("打开数据库错误!");
// 读取 XML 文件
if (!($fp = fopen($xml_file, "r")))
{
   die("打开文件错误: $xml_file");
}
// 解析 XML
while ($data = fread($fp, 4096))
{
   // 错误处理
   if (!xml_parse($xml_parser, $data, feof($fp)))
   {
      $ec = xml_get_error_code($xml_parser);
      die("XML 解析器错误(错误代号".$ec."):".xml_error_string($ec).
      "<br>发生错误的行".xml_get_current_line_number($xml_parser));
   }
}
xml_parser_free($xml_parser);
mysql_close($conn);
echo "导入 XML 数据成功";
?>
</body>
</html>
```

上述程序利用 SAX 解析 student.xml 文件。其中，利用 xml_parser_create()函数初始化 SAX 解析器实例，设定 SAX 解析器实例句柄，以调用各个回调函数，如 startElemen-Handler()、endElementHandler()。当解析 XML 文档的开始标记时，执行 startElemenHandler()函数。当解析 XML 文件的结束标记时，执行 endElementHandler() 函数，插入一个记录到 student 表。

需要给 startElemenHandler()、endElementHandler()函数传递标记名和解析器句柄作为参数。当处理 XML 文档的字符数据时，执行 characterDataHandler()函数，此时，需要为其指定字符数据（CDATA）文本作为参数。

SAX 解析器从 student.xml 文件中检索数据，将数据存储到一个相关的数组$values。SAX 解析器在找到元素（如 name、age、address）的字符数据后，检索数据，根据获取的 $values 数组中的数据，建立一个 SQL 的 INSERT 命令字符串。

运行该程序后，在 student 表增加两条记录，如下：

```
mysql> select * from student;
+--------+-----+--------------+
| name   | age | address      |
+--------+-----+--------------+
| 王超    |  25 | 广州市东风路   |
| 徐小帆  |  28 | 广州市天河     |
+--------+-----+--------------+
2 rows in set (0.00 sec)
```

15.2.2 用 DOM 导入数据到数据库

也可以利用 DOM 解析 XML 文档，把数据导入到数据库中。下面的例 15.5 的程序是用 DOM 解析 XML 文档，将数据导入到 mydb 数据库的 student 表的另一种方法。

【例 15.5】 用 DOM 解析 XML 文档，导入数据到数据库中（ex15_5.php）。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>用 DOM 导入 XML 文档到数据库</title>
</head>
<body>
<?php
$fields = array();
$values = array();
$xml_file="student.xml";    // 指定要解析的 XML 文件
$host = "localhost";        // 数据库连接参数
$user = "root";
$password = "123456";
$db = "mydb";
$xml_file = "student.xml";
$conn=mysql_connect($host,$user,$password) or die("不能连接 MySQL");
mysql_query("set names 'utf8'");
mysql_select_db($db) or die ("打开数据库错误!");
$dom = new DomDocument('1.0');
$dom->load($xml_file, LIBXML_NOBLANKS);
//取 XML 文档的根节点
$root = $dom->documentElement;
//读取根节点的属性,即表名
$table = $root->getAttribute("name");
//如果 XML 文档有记录数据元素
if ($root->hasChildNodes()) {
   $children = $root->childNodes;
   //对每个 record 元素,循环处理
   foreach($children as $recordnode)
   {
      $fieldnode = $recordnode->childNodes;
      //对一个 record 元素内的各个子元素,即字段,读取其名字和值
      foreach($fieldnode as $node) {
        $fields[] = $node->nodeName;  //字段名
        $values[] = $node->nodeValue;  //字段值
      }
      //构造一条 INSERT 命令,并执行 INSERT 命令
      $query = "INSERT INTO $table";
      $query .= "(" . join(", ", $fields) . ")";
```

```
        $query .= " VALUES(\"" . join("\", \"", $values) . "\");";
        mysql_query($query) or die ("查询错误: $query. " .mysql_error());
        // 复位内部计数器数和数组
        $fields = array();
        $values = array();
    }
}
mysql_close($conn);
echo "导入 XML 数据成功";
?>
</body>
</html>
```

15.3　用 SimpleXML 导入和导出数据

利用前面介绍的程序例子进行 XML 文档和 MySQL 数据库之间交换数据时，只能对数据库的一个表进行转换，而且必须预先在数据库中建立好该表的结构，这种程序缺乏通用性，不能对任何表的数据进行导入和导出。为了能够对任何一个数据库中的所有表与 XML 文档之间进行数据交换，下面给出的例 15.6 的程序就是一个通用的程序，可以把一个数据库中所有表的结构和记录内容转换为一个 XML 文档，也可以把一个 XML 文档的内容自动地转换为一个数据库。

【例 15.6】 MySQL 数据库和 XML 文档之间交换数据的通用程序（ex15_6.php）。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>使用 SimpleXML 扩展导入和导出数据库的数据</title>
</head>
<body>
<?php
//本程序已在中文数据库与 XML 文档之间交换数据成功,要求数据库表的编码为 utf8
// 定义 MySQL 和 XML 之间交换数据的类
class xml_mysqlconvert {
private $user;
private $pass;
private $host;
private $conn;
//(1)构建方法
public function __construct(){
    $num_args = func_num_args();
    if($num_args > 0){
        $args = func_get_args();
        $this->host = $args[0];
        $this->user = $args[1];
```

```
            $this->pass = $args[2];
            $this->connect();
        }
    }
    //(2)连接数据库的函数
    private function connect (){
        if (!$this->conn = mysql_connect($this->host,$this->user,$this->pass)){
            die("连接数据库错误".mysql_errno().": ".mysql_error());
            exit;
        }
    }
    //(3)打开数据库的函数
    public function selectdb($dbname){
      if (!mysql_select_db($dbname, $this->conn))
          die("打开数据库失败");
    }
    //(4)将 XML 转换为 MySQL 数据库
    public function xmltomysql($inputfile) {
        $conn = $this->connect();     //第 1 步,打开数据库
        //第 2 步打开 XML 文件,读取 XML
        if (!($file = fopen($inputfile,"r"))) die("打开 XML 文件错误");
        $xml = new SimpleXMLElement($inputfile,NULL,TRUE);
        //试图打开数据库.
        mysql_query("SET NAMES 'utf8'");
        try {
            if (mysql_query("CREATE DATABASE IF NOT EXISTS ".$xml->dbname."")){
            $this->selectdb($xml->dbname,$conn);  //选择要导出的数据库
            //创建表
          foreach ($xml->table as $table){ //对 XML 中每个 table 元素循环处理写入一个表
                //产生表结构定义
                $ctable="CREATE TABLE IF NOT EXISTS ".$table->tname." (";
                $colcount = 0;  //列号
                $totcolcount = 0;  //一个表的列数
                //求表的列数
                foreach ($table->tstructure->tcolumn as $totcol)
                $totcolcount++;
                foreach ($table->tstructure->tcolumn as $col){
                    $colcount++;
                    $ctable.= $col->Field." ";
                    $ctable.= $col->Type." ";  //处理 Null
                    if ($col->Null == "YES")
                      $ctable.= "NULL ";
                    else
                      $ctable.= "NOT NULL ";
                    //处理默认值
```

```
if($col->Default!= "") $ctable.="DEFAULT ".$col->Default." ";
    //处理 Auto_Increment 属性
    if($col->Extra!= "")  $ctable.= "AUTO_INCREMENT ";
    if($colcount!= $totcolcount){  //最后处理主键
        if ($col->Key == "PRI"){
          $ctable .= ",PRIMARY KEY(".$col->Field."), ";
        } else {
          $ctable .= ", ";
        }
    } else {
        if($col->Key == "PRI"){
          $ctable .= ",PRIMARY KEY(".$col->Field.") ";
        }
    }
}   //end of foreach 语句
$ctable .= ") ENGINE=MyISAM DEFAULT CHARSET=utf8;";
//试图创建表
try {
    if(mysql_query ($ctable)){
    //插入记录
        foreach($table->tdata->trow as $row){
            $insquery = "INSERT INTO ".$table->tname." (";
            //查找行号
            $totrow = 0;
            foreach($row->children() as $totchild)  $totrow++;
            //设置值的名字
            $currow = 0;
            foreach($row->children() as $name=>$node){
                $currow++;
                if ($currow != $totrow)
                  $insquery .= $name.", ";
                else
                  $insquery .= $name;
            }
            $insquery .= ") VALUES (";
            //插入字段值
            $currow = 0;
            foreach ($row->children() as $childrendata){
                $currow++;
                if ($currow != $totrow)
                  $insquery .= "'".$childrendata."', ";
                else
                  $insquery .= "'".$childrendata."'";
            }
            $insquery .= ")";
```

```
                        //执行插入记录命令
                        try {
                            if(!mysql_query($insquery)){
                             throw new exception(mysql_error());
                            }
                        } catch (exception $e) {
                            echo $e->getmessage ();
                        }
                    }
                } else {
                    throw new exception(mysql_error()."<br />");
                }
                } catch(exception $e) {
                    echo $e->getmessage ();
                }   //完成一个表的建立 end of foreach($table->tstructure ...)
            }  //end of foreach($xml->table...),完成所有表的建立
        } else {
          throw new exception (mysql_error());
        }   //ond of if(mysql_query(CREATE...))
    } catch(exception $e) {
        echo $e->getmessage ();
    }
    fclose($file);
    print "XML 文档数据已经被导入到 MySQL 数据库<br>";
}

//(5)将 mysql 转换为 xml 文档的函数
public function mysqltoxml($database,$outputfile) {
    //连接数据库
    $conn = $this->connect();
    //打开要导出的数据库
    $this->selectdb($database,$conn);
    mysql_query("SET NAMES 'gbk'");
    //以写方式打开 XML 文件
    if(!($file = fopen($outputfile,"wt"))) die("打开文件错误");
    //输出 XML 版本号
    fwrite($file, "<?xml version=\"1.0\" encoding='gb2312' ?>\n");
    //首先输出 XML 文档元素
    fwrite($file,"<db>\n");
    //输出根元素为数据库名
    fwrite($file,"\t<dbname>".$database."</dbname>\n");
    //遍历数据库,获取所有表名
    if ($tquery = mysql_query ("SHOW TABLES FROM $database")){
    if (mysql_num_rows ($tquery) > 0){
    while ($tdata = mysql_fetch_array ($tquery)){
```

```
fwrite ($file,"\t<table>\n");
fwrite ($file,"\t\t<tname>".$tdata[0]."</tname>\n");
//获取表的字段名
if ($fquery = mysql_query ("SHOW COLUMNS FROM ".$tdata[0]."")){
if (mysql_num_rows ($fquery) > 0){
//首先显示表结构
fwrite ($file,"\t\t<tstructure>\n");
$narr = array ();
while ($fdata = mysql_fetch_assoc ($fquery)){
    $narr[] = $fdata['Field'];
    fwrite($file,"\t\t\t<tcolumn>\n");
    fwrite($file,"\t\t\t\t<Field>".$fdata['Field']."</Field>\n");
    fwrite($file,"\t\t\t\t<Type>".$fdata['Type']."</Type>\n");
    fwrite($file,"\t\t\t\t<Null>".$fdata['Null']."</Null>\n");
    fwrite($file,"\t\t\t\t<Key>".$fdata['Key']."</Key>\n");
    fwrite($file,"\t\t\t\t<Default>".$fdata['Default']."</Default>\n");
    fwrite($file,"\t\t\t\t<Extra>".$fdata['Extra']."</Extra>\n");
    fwrite($file,"\t\t\t</tcolumn>\n");
}
fwrite($file,"\t\t</tstructure>\n");  //保存一个表的结构
//显示记录内容
if($dquery = mysql_query("SELECT * FROM ".$tdata[0]."")){
    if(mysql_num_rows($dquery) > 0 ){
        fwrite($file,"\t\t<tdata>\n");
        //一个表的记录集内容
        while($ddata = mysql_fetch_assoc ($dquery)){
            fwrite($file,"\t\t\t<trow>\n");
            $fcounter = 0;
            while($ele = each ($ddata)){  //一个记录的各字段内容
                fwrite ($file,"\t\t\t\t\t<".$narr[$fcounter].">".
                $ele['value']."</".$narr[$fcounter].">\n");
                $fcounter++;
            }
            fwrite($file,"\t\t\t</trow>\n");
        }
        fwrite($file,"\t\t</tdata>\n");
    }
} else {
    echo mysql_error();
}
}
} else {
echo mysql_error();
}
fwrite ($file,"\t</table>\n");
```

```
        }
        }
        } else {
          echo mysql_error();
        }
        fwrite($file,"</db>");
        fclose($file);
        print "MySQL 数据库数据已经被转换为 XML 文档<br>";
    }

    //(6)删除数据库的函数
    public function dropdb($thedb){
        if(!mysql_query("DROP DATABASE $thedb", $this->conn)){
            die("删除数据库错误: $thedb: <br>");
        }
    }
    //(7)关闭数据库连接的函数
    public function __destruct() {
        if (!mysql_close($this->conn)){
            die("关闭连接错误: <br/>0");
        }
    }
    }

    //主程序: 创建本类的实例对象
    $myconverter = new xml_mysqlconvert("localhost","root","123456");
    //将 mydb 数据库转换为 XML
    $myconverter->mysqltoxml("mydb","ex15_6.xml");
    //删除数据库 mydb
    $myconverter->dropdb("mydb");
    //创建数据库
    $myconverter->xmltomysql("ex15_6.xml");
    ?>
    </body>
    </html>
```

这个程序定义了一个 xml_mysqlconvert 类，该类定义了几个方法：__construct()是该类的构造方法，给类变量赋初值，并调用 connect()方法，建立与 MySQL 数据库服务器的连接。selectdb()方法打开指定的数据库。xmltomysql()方法将一个 XML 文档的数据导入到一个 MySQL 的数据库。mysqltoxml()将指定的一个数据库的所有表的结构和记录导出到一个 XML 文档。dropdb($thedb)方法删除指定的数据库。

在 mysqltoxml()方法中，接受一个 XML 文件名参数，指定要保存的 XML 文件位置。脚本程序扫描指定数据库中的每个表的结构和记录，生成 XML 的元素，并写入 XML 文件中。在创建的 XML 文档中，定义<db>元素为 XML 文档的根元素，<dbname>元素是<db>

元素的子元素，存放数据库名；每个<table>元素存放一个表的结构和记录，<tstructure>元素存储表结构，其每个字段由<tcolumn>子元素定义。<tdata>元素存储动态生成的记录集，其每个记录由一个<trow>子元素来定义。mysqltoxml()方法输出的 XML 文档形式如下：

```
<?xml version="1.0" encoding='gb2312'?>
<db>
    <dbname>mydb</dbname>
    <table>
        <tname>dept</tname>
        <tstructure>
            <tcolumn>
                <Field>deptID</Field>
                <Type>varchar(3)</Type>
                <Null>YES</Null>
                <Key></Key>
                <Default></Default>
                <Extra></Extra>
            </tcolumn>
            <tcolumn>
                <Field>deptName</Field>
                <Type>varchar(30)</Type>
                <Null>YES</Null>
                <Key></Key>
                <Default></Default>
                <Extra></Extra>
            </tcolumn>
        </tstructure>
        <tdata>
            <trow>
                    <deptID>001</deptID>
                    <deptName>行政科</deptName>
            </trow>
            <trow>
                    <deptID>002</deptID>
                    <deptName>生产科</deptName>
            </trow>
        </tdata>
    </table>
   ⋮
</db>
```

xmltomysql()方法是 mysqltoxml()方法的逆操作，该方法利用 SimpleXMLElement 类的构造方法 SimpleXMLElement($inputfile, NULL, TRUE)读取整个 XML 文档，然后使用 Simple XML 解析器来解析 XML 文档，得到数据库名，再解析<table>元素的数据，逐步地创建数据库以及每个表。

运行本程序，首先建立一个 XML 文档，存放 mydb 数据库所有表的结构和记录内容，然后删除 mydb 数据库，最后又根据创建的 XML 文档的内容，建立 mydb 数据库及其所有表。

通过这一个例子，可以看到利用 Simple XML 解析器和 MySQL 的功能，在 XML 文档和 MySQL 数据库两者之间交换数据不仅相当简便，而且功能强大。

需要注意的是，如果数据库的表数据含有中文，必须将表的字符集设置为 utf8，同时将本程序的文档编码也设置为 utf-8 格式，这样才能正确地导入和导出中文数据。

实　验　15

1．编写一个 PHP 程序，将 student_db 数据库的 score 表的记录内容导出到一个文件名为 score.xml 的 XML 文件。

2．编写一个 PHP 程序，将上一题的 score.xml 文件的内容导入到 student_db 数据库的 score1 表。

3．改写例 15.6 的程序，能够将 student_db 数据库的所有表导出到 students.xml 文件。

习　题　15

1．利用 DOM、SAX 和 SimpleXML 解析器来解析数据库的数据时，应该将数据库表的字符编码设置为什么样的编码，才能正确处理汉字？

2．理解例 15.6 的 xmltomysql()方法和 mysqltoxml()方法的程序代码。

第 16 章 用 PHP 开发的网络考试系统

本章导读

前面的各章介绍了 PHP 5 的技术和一些简单的应用，但没有涉及如何利用 PHP 来开发一个完整的网站实例。要开发一个网站，首先要对整个网站的布局进行规划，划分不同的功能结构，然后才能编程。本章通过一个网络考试系统应用实例，介绍如何综合利用 PHP、MySQL、Javascript 等知识来设计一个动态网站。

16.1 网络考试系统的需求分析和功能设计

随着计算机网络技术，特别是 Internet 技术的发展和普及，对高校的教学考试方式带来重大的影响。许多高校正在对考试方式进行改革，从传统的考试方式逐渐过渡到网络考试，或者两种考试方式并存。基于 Web 技术的网络考试系统可以借助于遍布全球的 Internet 进行，因此，网络考试既可以在本地进行，也可以在异地进行，考试形式更加灵活。

网络考试具有以下优点：

- 实现了无纸化考试，节约教学成本的开支。
- 使用计算机自动阅卷，大大提高阅卷效率。
- 试题存储在服务器，不容易泄露试题，试题的安全性和公平性得到有效的保障。
- 根据存储在服务器的考生答案数据，对考试质量进行定量分析，从而科学地评价学生掌握知识的程度。因此，网络考试将是考试改革的一种发展趋势。

在网络考试中，涉及三种不同的用户：学生、教师和管理员，他们的职能各不相同。学生进入网络考试系统，参加课程的考试，查看自己的成绩。教师能够在考试系统中添加试题、评阅学生答卷、提交成绩。管理员能够注册学生信息，管理教师信息，安排课程的考试时间等。所有这些数据都存储到服务器的数据库中。

在本章里，以 PHP 为编程脚本语言，MySQL 为后台数据库，Apache 为 Web 服务器，开发一个为教师提供手工命题、满足上述三种不同用户的要求的网络考试系统。

网络考试系统的功能结构如图 16.1 所示，下面简单介绍各个功能。

1．管理员功能部分

管理员负责对学生、教师身份，课程、班级、试题、考试时间进行全面的管理。其功能包括：

（1）课程管理。能够完成添加、删除和修改课程信息。

（2）班级管理。能够完成添加、删除和修改班级信息。

（3）学生管理。能够添加、删除和修改学生基本信息。为了在考试中能够核对学生身

份，还应提供学生照片的上传和显示。

（4）教师管理。能够添加、删除和修改教师登录信息。

（5）考试时间安排。安排课程在某天的某段时间进行考试，指定参加考试的班级。

（6）修改密码。管理员和教师都可以修改自己登录的密码。

（7）退出系统。管理员、教师和学生使用完考试系统后，执行退出功能，以清除相关数据。

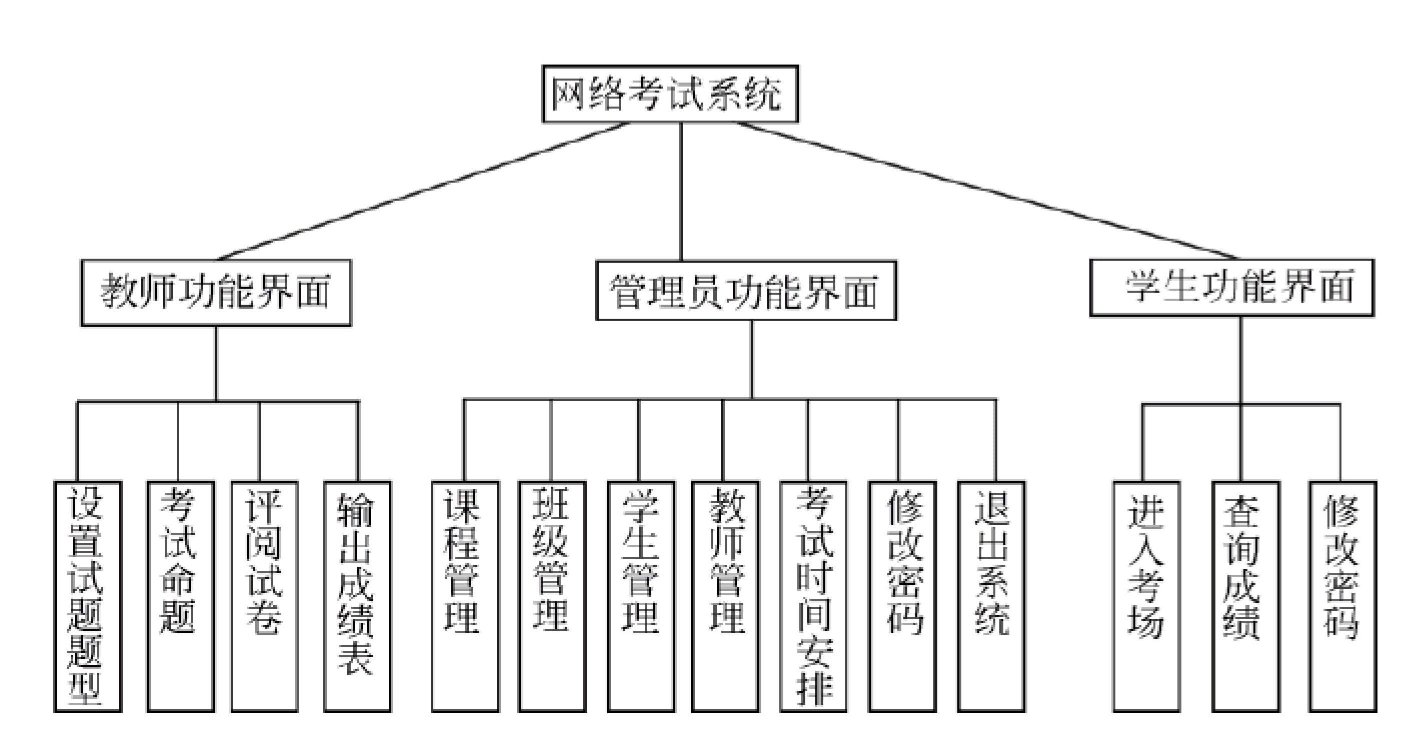


图 16.1　网络考试系统功能结构图

2．教师功能部分

教师的主要工作是完成试卷的命题和评阅答卷。其功能包括：

（1）设置试题题型。教师在给一门课程的试题输入题目之前，首先要添加一份试题，设置好该试题所包含的题型。

为了简单起见，在本考试系统中，只自动添加三种客观题型：单项选择题、多项选择题和判断题。读者可以在此基础修改程序，添加其他主观题型。

（2）考试命题。教师根据所选择的课程试题，给该份试题添加、修改和删除各种题型的题目。

（3）评阅试卷。教师根据所选择的试卷和班级，对一个班的学生答卷逐份进行评阅，生成学生的课程考试成绩。

（4）输出成绩表。教师根据所选择的课程和班级，输出一个班的课程成绩表。

3．学生功能部分

学生功能部分包括：

（1）进入考场。管理员安排好课程的考试时间后，学生在指定的时间前登录进入考试系统，使用“进入考场”功能，准备开始某一门课程的考试。当到达考试开始时间，自动地从服务器读取试题，传输到学生端的浏览器，学生即可答题。

（2）查询成绩。学生可以查询自己参加的各门课程的考试成绩。

16.2　数据库设计

在 MySQL 数据库系统中建立一个名为 exam_db 的数据库，存放考试系统中与课程、

学生、教师、试题、答卷相关的数据。该数据库包含以下数据表。

1. course 表

course 表存储各门课程的基本信息，其表结构如表 16.1 所示，主键为 course_id。

表 16.1 course 表结构

字段名	类型	是否 NULL	附加属性	含义
course_id	int(4)	NOT NULL	auto_increment	课程号
course_name	varchar(40)	NOT NULL		课程名称

2. teachuser 表

teachuser 表存储教师登录的信息，表结构如表 16.2 所示，主键为 t_id。

表 16.2 teachuser 表结构

字段名	类型	是否 NULL	附加属性	含义
t_id	int(5)	NOT NULL	auto_increment	序号
t_userid	varchar(20)	NOT NULL	unique key	教师用户名
t_username	varchar(20)	NOT NULL		教师姓名
t_pwd	varchar(45)			密码
t_usertype	varchar(20)			教师类别

说明：

（1）密码用 password("明密码")来加密。

（2）t_usertype 为教师类别，分为管理员和教师两种。

（3）第一次访问本系统时，自动添加管理员登录账号，用户名和密码都是 admin。

3. class 表

class 表存储班级信息，表结构如表 16.3 所示，主键为 class_id。

表 16.3 class 表结构

字段名	类型	是否 NULL	附加属性	含义
class_id	int(10)	NOT NULL	auto_increment	班级序号
enroll_year	int(4)	NOT NULL		入学年度
classname	varchar(30)	Not null		班级名称

4. student_user 表

student_user 表存储学生的基本信息，表结构如表 16.4 所示，主键是 s_id。

表 16.4 student_user 表结构

字段名	类型	是否 NULL	附加属性	含义
s_id	int(12)	Not null	auto_crement	学生序号
s_xh	varchar(16)	Not null	Unique key	学号
s_name	varchar(14)	Not null		姓名
s_pwd	varchar(45)	Not null		密码
s_class_id	int(10)	NOT NULL		班级序号
s_photo	mediumblob	null		照片

5．exam_type 表

exam_type 表存储各门课程的试卷题型，其表结构如表 16.5 所示，主键为 id。

表 16.5　exam_type 表结构

字段名	类型	是否 NULL	附加属性	含义
id	int(4)	not null	auto_increment	序号
exam_id	varchar(26)	not null		试卷编号
exam_type_id	int(4)	NOT NULL		题型号
exam_type_name	varchar(30)	NOT NULL		题型名称
exam_type_desc	text	null		题型描述
auto_grade	int(1)	NOT NULL	0	是否自动评分

说明：

（1）auto_grade 列的值为 1（自动评分）或 0（人工评卷）。

（2）当添加一份试卷时，自动添加三种题型：单项选择题、多项选择题、判断题。

6．exam_info 表

该表存储试卷基本信息，表结构如表 16.6 所示。

表 16.6　exam_info 表结构

字段名	类型	是否 NULL	附加属性	含义
course_id	int(4)	NOT NULL		课程号
exam_id	varchar(26)	Not null	Unique key	试卷编号
exam_title	varachr(250)	NOT NULL		试卷名称
exam_header	text	null		试卷头部标题
exam_t_userid	varchar(20)			命题教师用户名
exam_t_name	varchar(20)			教师姓名
exam_prop_date	date			命题日期
exam_audit	int(1)			是否已提交试题

7．exam_score 表

该表存放学生各门课程的成绩，表结构如表 16.7 所示，主键是 id。

表 16.7　exam_score 表结构

字段名	类型	是否 NULL	附加属性	含义
id	int(12)	not null	auto_increment	序号
s_xh	char(16)	not null		学号
course_id	int(4)	not null		课程号
exam_id	varchar(26)	not null		试卷编号
score	decimal(7,1)	null		课程成绩

8．exam_test 表

该表存储每份试题的题目、标准答案和标准分值，表结构如表 16.8 所示，主键是 st_id。

表 16.8　exam_test 表结构

字段名	类型	是否 NULL	附加属性	含义
st_id	int(12)	not null	auto_increment	题目序号
exam_id	varchar(26)	Not null	Unique key	试卷编号
tk_type_id	int(4)	NOT NULL		题型号
tk_content	text			题目内容或标题
tk_option1	text			题目选项 1
tk_option2	text			题目选项 2
tk_option3	text			题目选项 3
tk_option4	text			题目选项 4
tk_option5	text			题目选项 5
tk_option6	text			题目选项 6
tk_pic	mediumblob			题目插图
tk_ans	text			答案
tk_ans_pic	mediumblob			答案插图
standard_score	Decimal(7,1)			标准分值
auto_grade	Int(1)		默认值 0	是否自动评分

9．exam_time 表

该表存储各门课程的考试时间，表结构如表 16.9 所示，主键是 exam_id。

表 16.9　exam_time 表结构

字段名	类型	是否 NULL	含义
course_id	int(4)	NOT NULL	课程号
exam_id	varchar(26)	Not null	试卷编号
exam_class	text		考试班级号
exam_date	date		考试日期
exam_starttime	time		考试开始时间
exam _endtime	time		考试结束时间
exam_timelen	int		考试时长（分钟）

10．stud_exam_ans 表

该表存储学生的答卷内容，表结构如表 16.10 所示，主键是 id。

表 16.10　stud_exam_ans 表结构

字段名	类型	是否 NULL	附加属性	含义
id	Int(14)	Not null	auto_increment	序号
s_xh	Char(16)	Not null		学号
course_id	int(4)	NOT NULL		课程号
exam_id	Int(12)	Not null		试卷编号
st_id	Int(12)	Not null		题目序号
stud_ans	text	null		学生答案
stud_ans_pic	blob	null		学生答案插图
stud_score	decimal(7,1)			该题学生得分
t_userid	varchar(14)			评卷教师姓名

16.3　全局变量和公共模块

网络考试系统的工作流程如图 16.2 所示。首先显示主页，然后进入登录页面，如图 16.3 所示。在登录页面中，选择用户类别，输入用户名（或学号）和密码，进行用户身份的合法性验证。如果用户合法，则进入相应的页面，如图 16.4、图 16.5 和图 16.6 所示。

网站的目录结构是：admin 子目录存放教师和管理员执行的所有程序，exam 子目录存放学生执行的所有程序，js 子目录存放外部的 CSS 文件和 JavaScript 文件。

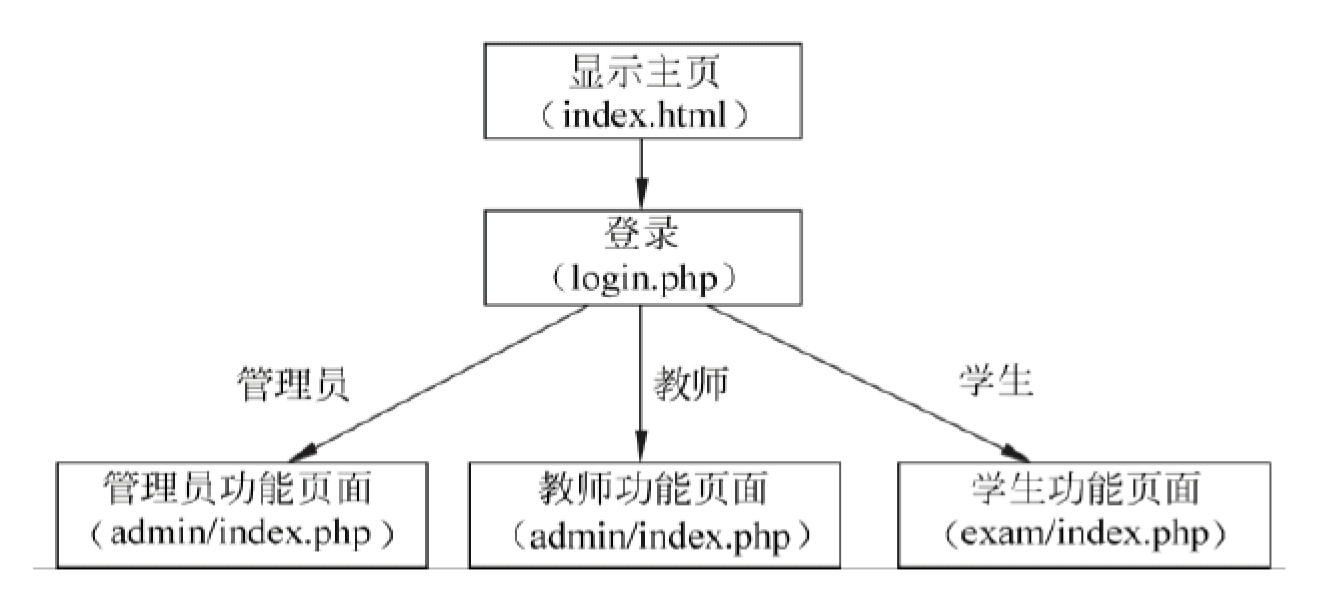


图 16.2　网络考试系统工作流程示意图



图 16.3　登录页面

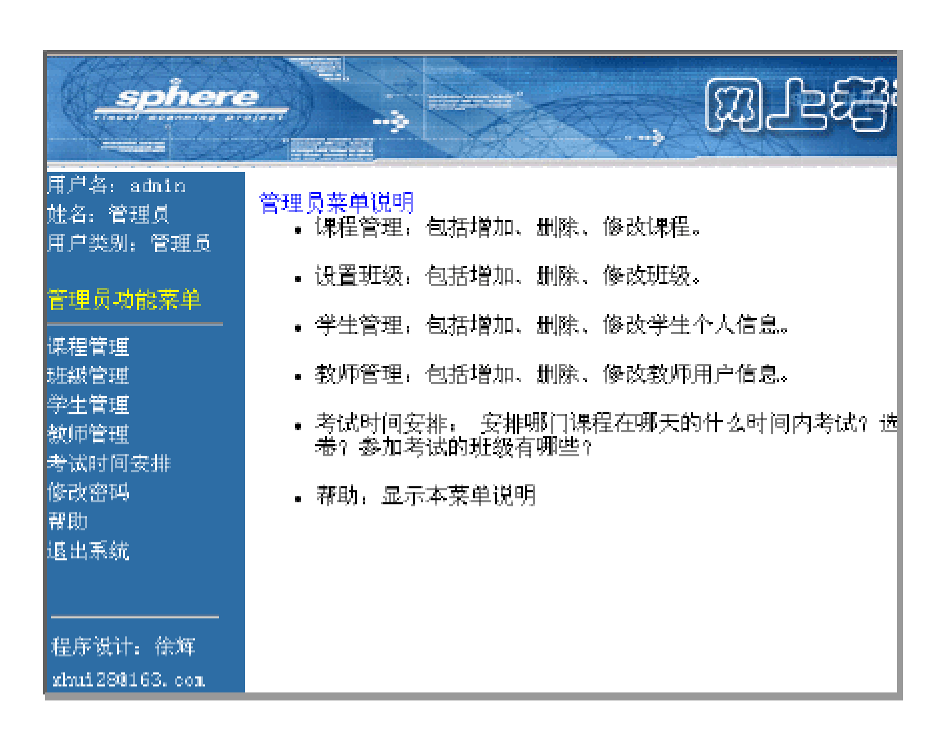


图 16.4　管理员功能页面

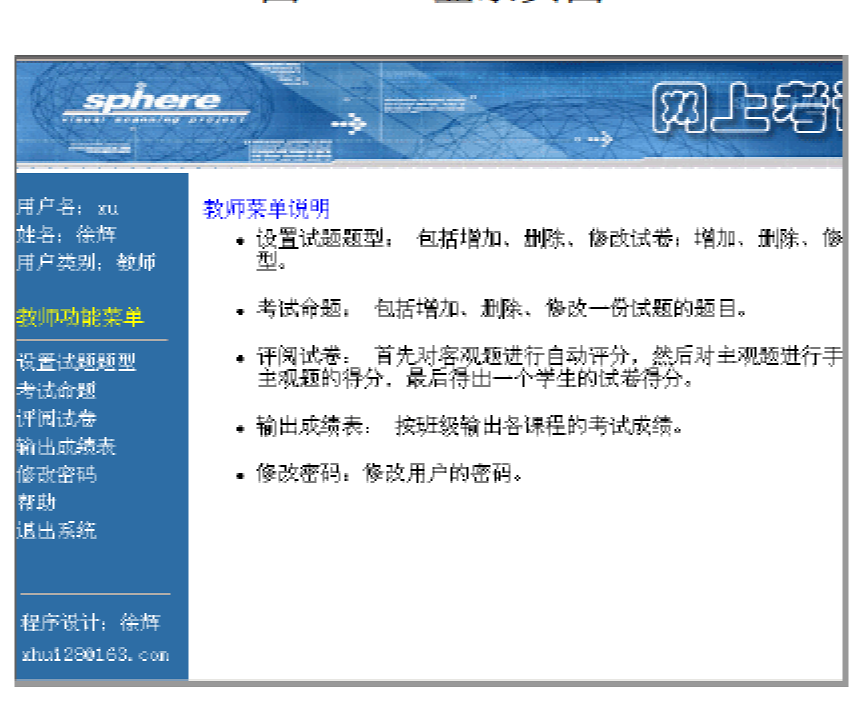


图 16.5　教师功能页面

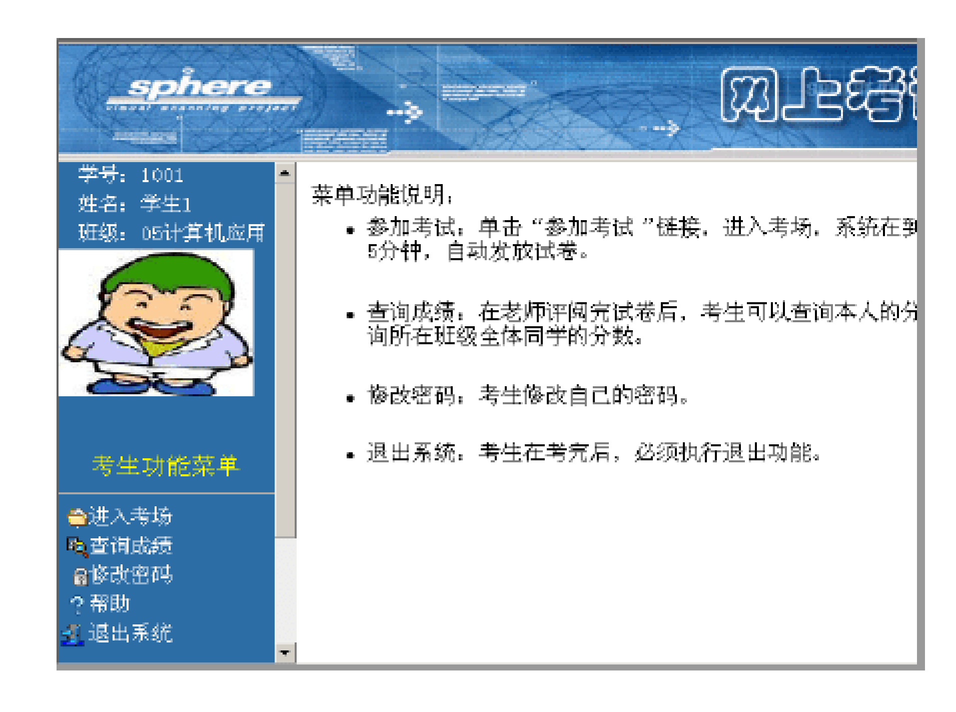


图 16.6　学生功能页面

16.3.1 全局变量

为了在考试系统的所有程序中获取当前用户的信息，采用 SESSION 变量存储用户登录的信息。这些 SESSION 变量为$_SESSION["xh"]、$_SESSION["name"]、$_SESSION["classname"]分别存放学生的学号、姓名、班级名称。$_SESSION["t_userid"]、$_SESSION["name"]分别存放教师或管理员的用户名和姓名。因此，在每一个 PHP 脚本程序的开头，必须加入以下语句，以便引用 SESSION 变量。

```
session_start();
```

16.3.2 公共模块

1．连接 MySQL 服务器程序（conn.php）

该程序建立与 MySQL 服务器的连接，打开 exam_db 数据库。程序中的$host、$user、$passwd 变量的值应根据所使用的 MySQL 服务器而更改。代码如下：

```
<?
$host="localhost";
$user="root";
$passwd="123456";
$db="exam_db";
$conn=mysql_connect($host,$user,$passwd);
mysql_selectdb($db);
mysql_query("set names 'gbk'");
?>
```

在每一个 PHP 程序的开头，通过以下语句引用 conn.php 程序。

```
require("../conn.php");
```

2．退出系统程序（logout.php）

退出程序用来删除 SESSION 变量，返回主页面，以便其他用户能够登录系统。代码如下：

```
<?
session_start();
session_destroy();
$_SESSION=array();
?>
<script language="JavaScript">
top.location.href="./index.html";
</script>
```

16.4 管理员功能的程序

根据设计的功能结构和数据库结构，从本节起，介绍一些主要功能的实现过程和程序，其余功能的实现程序参见课件。

16.4.1　课程管理

课程管理程序（admin_course.php）实现增加、删除和修改课程信息，将课程信息保存到 course 表。其页面分为两部分：上面部分是一个表单，用来增加课程，下面部分显示当前已定义的课程。页面运行效果如图 16.7 所示。主要的程序代码如下：

课程名称		添加课程

课程编号	课程名称	操作
1	Visual Foxpro数据库应用	编辑 删除
13	VB程序设计	编辑 删除
14	操作系统	编辑 删除

图 16.7　课程管理页面

```
<?
 session_start();
require("../conn.php");
$sql="select * from course";
$rs=mysql_query($sql,$conn) or die("查询课程失败");
?>
<form action="admin_course_insert.php" method="POST" name="courseform1">
  <table border="1">
     <tr><td >课程名称</td>
          <td ><input type="text" name="course_name">
          <input border="0" value="添加课程" type="submit">
      </td></tr>
  </table>
</form>
  <table border="1">
    <tr> <td>课程编号</td> <td>课程名称</td> <td >操作</td>
    </tr>
    <?
    $i=0;
    while ($row=mysql_fetch_array($rs)) {
    ?>
    <tr>
      <input name="course_id" type="hidden" value="<? echo $row
      ['course_id']; ?>">
      <td><? echo $row['course_id'];?></td>
      <td><? echo $row["course_name"];?></td>
      <td align="center"> <a href="admin_course_edit.php?id=<? echo $row
      ['course_id']; ?>">编辑</a>
  <a href="admin_course_del.php?id=<? echo $row['course_id'];?>">删除
  </a></td>
  </tr>
  <?
  $i++;
  }
  ?>
</table></body></html>
```

单击“添加课程”按钮，提交表单数据，调用 admin_ccourse_insert.php 程序，完成课程记录的插入。admin_ccourse_insert.php 程序内容如下：

```
<?
session_start();
require("../conn.php");
$sql="insert into course(course_name) values('{$_POST['course_name']}')";
mysql_query($sql,$conn);
header("Location:./admin_course.php");
?>
```

“编辑”操作通过超链接，定位到 admin_course_edit.php 程序，由该程序完成修改课程。“删除”操作通过超链接，定位到 admin_course_del.php 程序，删除指定的课程。这两个程序的代码比较简单，在此略述，详见课件。

16.4.2 班级管理

班级管理程序（admin_class.php）实现增加、删除和修改班级信息，将班级信息保存到 class 表。其页面分为两部分：上面部分是一个表单，用来增加班级；下面部分显示当前已经设置的班级。页面运行效果如图 16.8 所示。其代码与课程管理程序类似，操作也与课程管理的相似。

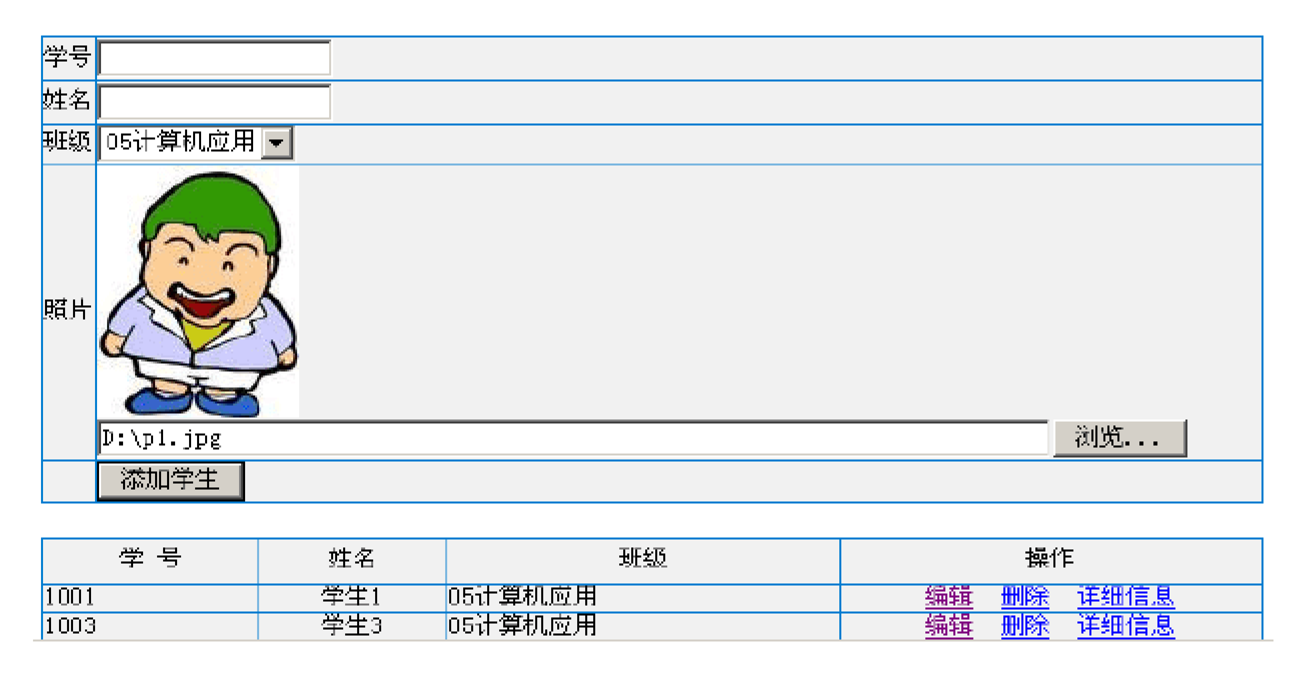
入学年度	
班级名称	添加班级

班级编号	入学年度	班级名称	操作
1	2005	05计算机应用	编辑 删除
2	2006	06网络	编辑 删除
7	2006	06信息管理	编辑 删除

图 16.8 班级管理页面

16.4.3 学生管理

学生管理程序实现增加、删除和修改学生个人信息，将学生个人信息保存到 student_user 表。其页面分为两部分：上面部分是一个增加学生的表单，下面部分显示当前已经注册的学生。页面运行效果如图 16.9 所示。



学号	
姓名	
班级	05计算机应用
照片	D:\p1.jpg 浏览...
	添加学生

学 号	姓名	班级	操作
1001	学生1	05计算机应用	编辑 删除 详细信息
1003	学生3	05计算机应用	编辑 删除 详细信息

图 16.9 学生管理页面

1．学生管理主程序（admin_student.php）

在 admin_student.php 程序中，表单部分的主要代码如下：

```
<form action="admin_student_insert.php" method="POST" enctype=
"multipart/form-data" name="studentform1" >
  <table border="1">
    <tr><td>学号</td><td><input name="xh" type="text"></td>
    </tr>
    <tr><td>姓名</td><td><input name="name" type="text"></td>
    </tr>
    <tr><td >班级</td>
      <td ><select name="class_id" size="1">
      <?
         $rs2=mysql_query("select * from class",$conn);
         if (mysql_num_rows($rs2)>0) {
           $row=mysql_fetch_array($rs2);
          echo "<option value=".$row['class_id']." selected>".
     $row['classname']."</option>";
           while ($row=mysql_fetch_array($rs2)) {
              echo "<option value=".$row['class_id'].">".
              $row['classname']."</option>";
           }
          }
     ?>
    </select></td>
    </tr>
    <tr>
      <td>照片</td>
      <td><img src="" name="myphoto">
        <input name="photofile" type="file" onChange="studentform1.
        myphoto.src=this.value;">
         <input type="hidden" name="MAX_FILE_SIZE" value="2097152" />
    </td>
    </tr>
    <tr>
      <td height="23"  align="center"> </td>
      <td ><input name="submit" type="submit" id="submit2" value="添加学生"
      border="0"></td>
    </tr>
  </table>
</form>
```

针对这个表单，应该注意的是，需要将表单的 enctype 属性值设置为"multipart/form-data"，才能上传图片。为了让管理员在选择一个图片文件后，能够即时看到所选的图片（参

见图 16.9），在表单中插入一个<img>标记：<img src="" name="myphoto">。同时在文件域标记中设置 onChange 事件代码，使得<img>标记的图形来自于所选择的图形文件，如下：

```
<input name="photofile" type="file" onChange="studentform1.
myphoto.src=this.value;">
```

2．保存新增加的学生信息的程序

在图 16.9 中，单击“添加学生”按钮，提交表单的数据到 Web 服务器，由 admin_student_insert.php 程序接收表单的数据，把学生信息作为一个记录插入到 student_user 表。该程序内容如下：

```
<?
session_start();
require("../conn.php");
$name=$_FILES['photofile']['tmp_name'];  //上传的照片文件名
if (! empty($name)) {
    $photo=fread(fopen($name,"rb"),filesize($name));
    $photo = '0x' . bin2hex($photo);
    $sql ="insert into student_user(s_xh,s_name,s_pwd,";
    $sql.="s_class_id,s_photo) values('{$_POST['xh']}',";
    $sql.=" '{$_POST['name']}',PASSWORD('{$_POST['xh']}'),";
    $sql.="{$_POST['class_id']},$photo)";
}else {
    $sql="insert into student_user(s_xh,s_name,s_pwd,";
    $sql.="s_class_id) values('{$_POST['xh']}',";
    $sql.="'{$_POST['name']}',PASSWORD('{$_POST['xh']}'),";
    $sql.="{$_POST['class_id']})";
}
mysql_query($sql,$conn);
header("Location:./admin_student.php);
?>
```

程序中，利用 fread()函数以二进制方式读取上传的图形文件内容，然后用 bin2hex()将二进制数表示的图形转换为十六进制数，在 INSERT 命令中使用十六进制数表示照片内容。

16.4.4 教师管理

教师管理程序（admin_teacher.php）实现增加、删除和修改教师信息，将教师信息保存到 teachuser 表。程序内容与学生管理的程序类似。

其页面运行效果如图 16.10 所示。单击“编辑”链接，则调用 admin_teacher_edit.php 程序，修改指定教师的信息。单击“删除”链接，删除指定教师的信息。

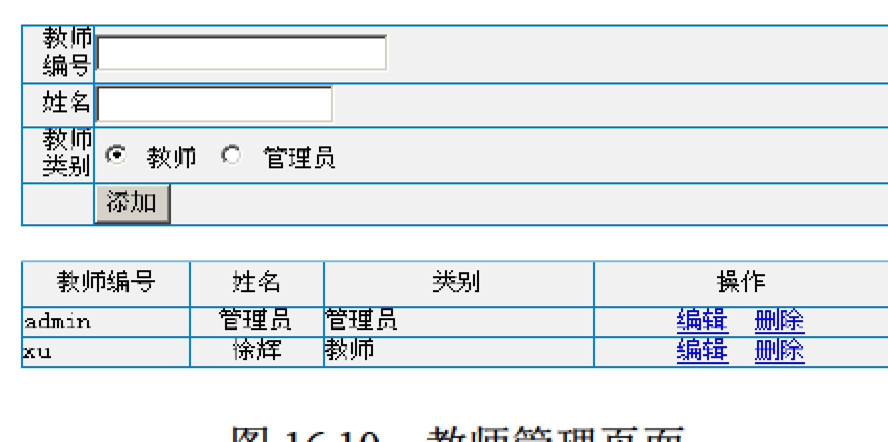


图 16.10 教师管理页面

16.4.5 考试时间安排

考试时间安排程序（admin_exam_time.php）用来设置每门课程的考试日期和时间。其运行界面如图 16.11 所示。它是一个表单，为了在“试卷名称”下拉列表中选择一份试卷后，能够在“考试课程”、“试卷编号”、“命题教师”这三个文本框中显示与试卷相应的内容，利用 PHP 和 JavaScript 结合编程，产生与这些表单元素对应的数组，代码如下：

设置课程的考试时间			
试卷名称	《VFP数据库应用》课程试卷（A）		
考试课程	Visual Foxpro数据库应用		
试卷编号	exam20070131085924		
命题教师	徐辉		
考试班级	☐ 05计算机应用 ☐ 06网络 ☐ 06信息管理		
考试时间设定			
考试日期	（日期格式：yyyy-mm-dd。如2007-01-12表示2007年1月12日）	考试时长	（分钟）
考试开始时间	（时间格式：hh:mm。如15:00表示15时0分）		
考试结束时间	（时间格式：hh:mm。如15:00表示15时0分）		
保存			

图 16.11　考试时间安排页面

```
<?
session_start();
require("../conn.php");
$sql ="select * from exam_info , course ";
$sql.="where  exam_info.course_id=course.course_id";
$rs=mysql_query($sql,$conn);
$examid=$examname=$courseid=$coursename=$teacher="";
while ($row=mysql_fetch_array($rs)) {
    $examid.='"'.$row["exam_id"].'",';
    $examtitle.='"'.$row["exam_title"].'",';
    $courseid.=$row["course_id"].',';
    $coursename.='"'.$row["course_name"].'",';
    $teacher.='"'.$row["exam_t_name"].'",';
}
if (strlen($examid)>0) {
   $examid=substr($examid,0,strlen($examid)-1);
   $examtitle=substr($examtitle,0,strlen($examtitle)-1);
   $courseid=substr($courseid,0,strlen($courseid)-1);
   $coursename=substr($coursename,0,strlen($coursename)-1);
   $teacher=substr($teacher,0,strlen($teacher)-1);
}
?>
<SCRIPT LANGUAGE="JavaScript">
function select_exam(i) {
<?
```

```
echo "     var examid = new Array(".$examid.");\n";
echo "     var examtitle = new Array(".$examtitle.");\n";
echo "     var courseid = new Array(".$courseid.");\n";
echo "     var coursename = new Array(".$coursename.");\n";
echo "     var teacher = new Array(".$teacher.");\n";
?>
    var id,i;
    document.form1.coursename.value=coursename[i];
    document.form1.courseid.value=courseid[i];
    document.form1.examid2.value=examid[i];
    document.form1.exam_t_name.value=teacher[i];
}
</SCRIPT>
```

这段程序从 exam_info 和 course 表中读取每份试卷的名称、编号及课程名、命题教师，动态生成 JavaScript 脚本的数组。Javascript 的 select_exam()函数用来改变表单中显示的考试课程、试卷编号、命题教师这三个文本框的值。

为了在表单的“试卷名称”下拉列表中选择一份试卷后，能够触发 select_exam()函数，还需要给该下拉列表标记定义 onChange 事件代码，如下：

```
<select name="examid" onChange="select_exam(this.selectedIndex)">
```

在图 16.11 所示的表单中，输入考试时间后，单击“保存”按钮，提交表单，调用 admin_examtime_save.php 程序，将考试时间信息保存到 exam_time 表中，并将参加考试的班级学生信息添加到 exam_score 表。admin_examtime_save.php 程序如下：

```
<?
session_start();
require("../conn.php");
//将表单的班级号数据转换为字符串,用逗号分开
$class=implode(",", $_POST["exam_class"]);
//生成 INSERT 命令
$sql="insert into exam_time(course_id, exam_id, exam_class,";
$sql.="exam_date,exam_starttime,exam_endtime,";
$sql.="exam_timelen) values({$_POST['courseid']},";
$sql.="'{$_POST['examid']}','$class',";
$sql.="'{$_POST['exam_date']}',";
$sql.="'{$_POST['exam_starttime']}',
$sql.="'{$_POST['exam_endtime']}',";
$sql.="'{$_POST['exam_timelen']}')";
$rs=mysql_query($sql,$conn);
//以下循环将各班参加考试的学生信息添加到成绩表 exam_score 表
$class=$_POST["exam_class"];
for ($i=0;$i<count($class);$i++) {
   $sql ="select * from student_user ";
```

```
    $sql.=" where s_class_id=".$class[$i];
    $rs=mysql_query($sql,$conn);
    $sql="insert into exam_score(s_xh,course_id,exam_id) values ";
   while ($row=mysql_fetch_array($rs))
     $sql.="('{$row['s_xh']}',{$_POST['courseid']},";
     $sql.="'{$_POST['examid']}'),";
     $sql=substr($sql,0,strlen($sql)-1);
     mysql_query($sql,$conn);
}
header("Location:admin_exam_time.php");
?>
```

16.5 教师功能的程序

16.5.1 设置试题题型

设置试题题型程序用来完成试题基本信息的添加、删除和修改，并设置试题的题型。

1. 设置试题题型主程序（teach_exam_type_step1.php）

此程序显示当前已经设置的试卷基本信息，如图 16.12 所示。程序如下：

```
<?
//文件名: teach_exam_type_step1.php
session_start();
require("../conn.php");
$sql ="select * from exam_info,course ";
$sql.="where exam_t_userid='{$_SESSION['t_userid']}' ";
$sql.="and exam_audit=0 and ";
$sql.="exam_info.course_id=course.course_id";
$rs=mysql_query($sql,$conn);
?>
<script language="javascript">
function redirectit(){
   self.location.href="teach_exam_add.php";
}
</script>
<body>
```

设置试卷基本信息：添加、修改试卷名；添加、删除、修改试卷的题型。

```
<table border="1">
  <tr><td colspan="5" align="left">第一步: 选择试卷</td></tr>
  <tr><td>课程名称</td><td >试卷名称</td>
    <td>命题教师</td><td>命题日期</td><td>操作</td>
  </tr>
  <?
```

```
while ($row=mysql_fetch_array($rs)) {
?>
 <tr>
  <td ><? echo $row['course_name'];?></td>
  <td><a href="teach_exam_type_step2.php?examid=<? echo $row['exam_id'];
  ?>"><? echo $row['exam_title'];?></a></td>
  <td align="center"><? echo $row['exam_t_name'];?></td>
  <td align="center"><? echo $row['exam_prop_date'];?></td>
  <td align="center"><a href="teach_exam_edit.php?examid=<? echo $row
  ['exam_id'];?>">编辑</a> 
   <a href="teach_exam_del.php?examid=<? echo $row['exam_id'];?>" target=
  "_self">删除</a></td>
 </tr>
 <?
 }
 ?>
 <tr><td colspan="5">
<input type="submit" name="Submit" value="添加试卷" onclick="redirectit()">
</td>
</tr>
</table>
```

运行界面如图 16.12 所示。单击“添加试卷”按钮，显示图 16.13 所示的页面。

第一步：选择试卷

课程名称	试卷名称	命题教师	命题日期	操作
Visual Foxpro数据库应用	《VFP数据库应用》课程试卷（A）	徐辉	2007-01-03	编辑 删除
VB程序设计	《Visual Basic程序设计》试卷（A）	徐辉	2007-01-08	编辑 删除
				添加试卷

图 16.12 设置试题题型

2．添加试卷程序（teach_exam_add.php）

此程序显示一个表单，如图 16.13 所示，输入试卷信息，单击“添加试卷”按钮，提交表单数据，再次调用本程序，将试卷信息添加到 exam_info 表。程序如下：

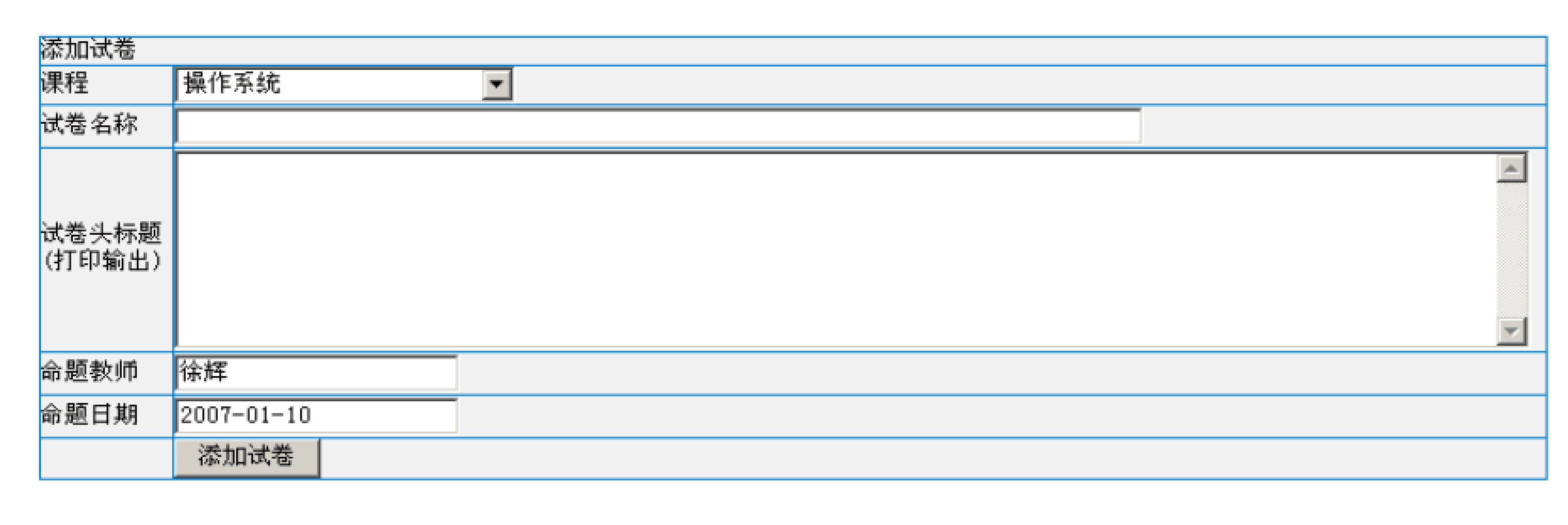


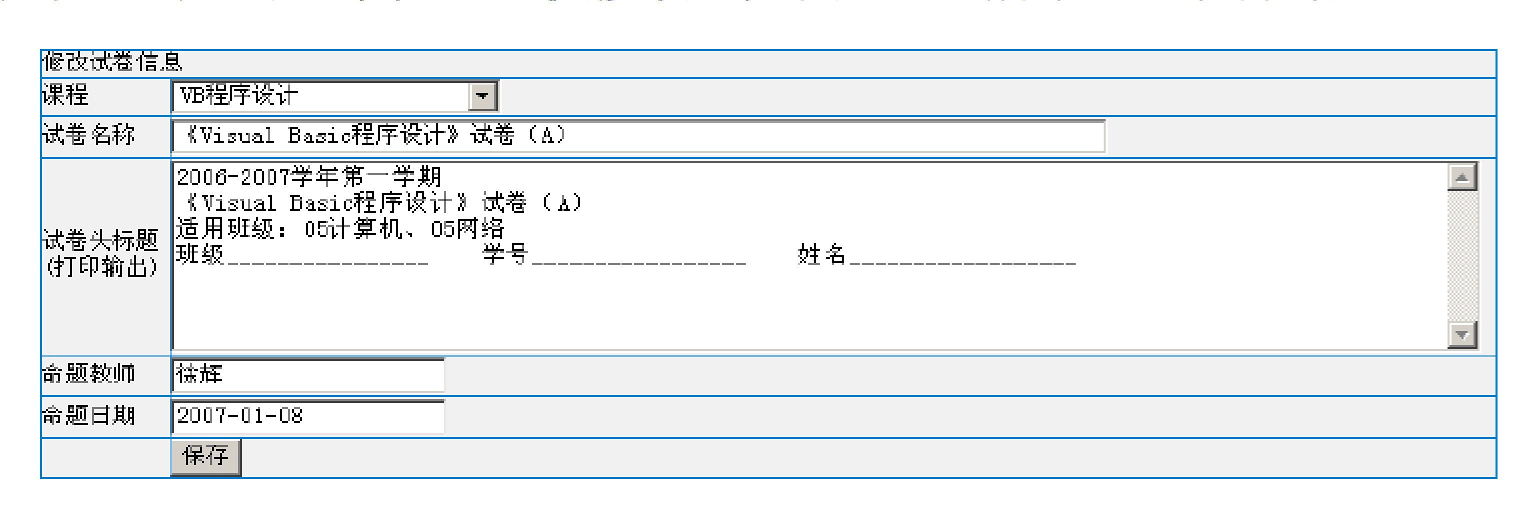
图 16.13 添加试卷表单

```
<?
session_start();
require("../conn.php");
if ($_POST['Submit']=="添加试卷") {
  $exam_id="exam".date("YmdHis");  //试卷编号
  $examdate=date("Y-m-d");
  $sql ="insert into exam_info (course_id,exam_id,exam_title,";
$sql.="exam_header,exam_t_userid,exam_t_name,exam_prop_date)";
  $sql.="values({$_POST['course_id']},'{$exam_id}',";
  $sql.="'{$_POST['exam_title']}','{$_POST['exam_header']}'";
  $sql.=",'{$_SESSION['t_userid']}','{$_SESSION['name']}',";
  $sql.="'{$examdate}')";
   mysql_query($sql,$conn);
   header("Location:./teach_exam_type_step1.php");
   exit;
}
?>
<body>
<form method="POST" action="teach_exam_add.php">
  <table border="1"><tr><td colspan="2">添加试卷</td></tr>
    <tr><td>课程</td><td><select name="course_id" >
     <?
      $rs2=mysql_query("select * from course");
      while ($row2=mysql_fetch_array($rs2)){
         echo "<option value='".$row2['course_id']."'>";
         echo $row2['course_name']."</option>";
      }
     ?>
    </select>
    </td></tr>
    <tr><td >试卷名称</td><td>
    <input name="exam_title" type="text"></td> </tr>
    <tr><td height="113" >试卷头标题<br>(打印输出)</td><td>
    <textarea name="exam_header" cols="100" rows="7"></textarea>
    </td>
    </tr>
    <tr><td height="18" >命题教师</td>
      <td><input name="exam_teacher" type="text" value="<? echo $_SESSION
      ['name'];?>" size="20" readonly></td>
    </tr>
    <tr><td>命题日期</td>
      <td><input name="exam_prop_date" type="text" value="<? echo $examdate=
      date("Y-m-d");?>" size="20" readonly></td>
    </tr>
    <tr><td> </td>
```

```
    <td ><input type="submit" name="Submit" value="添加试卷"></td>
     </tr>
   </table>
</form>
```

3．编辑试卷程序（teach_exam_edit.php）

在图 16.12 所示的页面中，单击某一个“编辑”超链接，则调用 teach_exam_edit.php 程序，显示某一试卷的基本信息，供修改，如图 16.14 所示（此程序略）。



修改试卷信息	
课程	VB程序设计
试卷名称	《Visual Basic程序设计》试卷（A）
试卷头标题(打印输出)	2006-2007学年第一学期 《Visual Basic程序设计》试卷（A） 适用班级：05计算机、05网络 班级_______________ 学号_______________ 姓名_______________
命题教师	徐辉
命题日期	2007-01-08
	保存

图 16.14　编辑试卷信息表单

4．设置试题的题型（teach_exam_type_step2.php）

在图 16.12 所示的页面中，单击某一个试卷名称超链接，则调用 teach_exam_type_step2.php 程序，自动添加三种客观题型，如图 16.15 所示（此程序略）。

第二步：设置试卷的题型

课程名称	试卷名称	命题教师
VB程序设计	《Visual Basic程序设计》试卷（A）	徐辉

题型序号	题型名称	操作
1	单项选择题（60分）	编辑
2	多项选择题（30分）	编辑
3	判断题（10分）	编辑

说明：序号为1、2、3的题型为客观题，可以自动评分。请保留这三种题型。在命题时，如果没有某题型的题目，不需要增加该题型的题目。

图 16.15　设置试卷题型页面

5．修改题型程序（teach_exam_type_edit.php）

在图 16.15 所示的页面中，单击某一题型的“编辑”超链接，调用 teach_exam_type_edit.php，显示图 16.16 所示的表单，修改题型的说明（此程序略）。

修改题型	
题型序号	1
题型名称	单项选择题（60分）
题型描述	第小题2分，共60分
是否自动评分	☑
	保存题型

图 16.16　编辑题型信息页面

16.5.2　考试命题

考试命题的功能是给某一份试题输入题目、修改题目和删除题目，它包括以下几个

程序。

1．显示试卷名称的程序（teach_examtest_step1.php）

这是考试命题的第一步，显示出当前已经定义的试卷名，如图 16.17 所示。程序代码如下：

第一步：选择试卷			
课程名称	试卷名称	命题教师	命题日期
Visual Foxpro数据库应用	《VFP数据库应用》课程试卷（A）	徐辉	2007-01-03
VB程序设计	《Visual Basic程序设计》试卷（A）	徐辉	2007-01-08

图 16.17　显示试卷名的页面

```
<?
session_start();
require("../conn.php");
$sql="select * from exam_info,course ";
$sql.=" where exam_t_userid='{$_SESSION['t_userid']}' and exam_audit=0 and ";
$sql.="exam_info.course_id=course.course_id";
$rs=mysql_query($sql,$conn);
?>
<title>考试命题</title>
<body>
<table border="1">
  <tr> <td colspan="4" >第一步: 选择试卷</td></tr>
  <tr><td>课程名称</td><td>试卷名称</td>
      <td>命题教师</td><td>命题日期</td>
  </tr>
  <?
 while ($row=mysql_fetch_array($rs)) {
 ?>
  <tr>
    <td><? echo $row['course_name'];?></td>
    <td><a href="teach_examtest_step2.php?examid=<? echo $row
    ['exam_id'];?>"><? echo $row['exam_title'];?></a></td>
    <td align="center"><? echo $row['exam_t_name'];?></td>
    <td align="center"><? echo $row['exam_prop_date'];?></td>
  </tr>
  <?
  }
  ?>
</table></body>
```

2．显示试题的程序（teach_examtest_step2.php）

在图 16.17 所示的页面中，单击某一个试题名称，显示该试题的内容，如图 16.18 所示，显示当前已经添加的试题题目，供修改、删除和增加（此程序略）。

3．增加单项选择题目的程序（teach_examtest_add_singlechoice.php）

该程序根据所选试卷编号为参数，显示一个表单，输入题目内容、选项和答案，如果

题目有插图，可以选择图形文件。程序内容如下：

第二步：编辑试题	
课程名称	VB程序设计
试卷名称	《Visual Basic程序设计》试卷（A）
试卷标题	2006-2007学年第一学期 《Visual Basic程序设计》试卷（A） 适用班级：05计算机、05网络 班级_______________ 学号________________ 姓名_________________
命题教师	徐辉
命题日期	2007-01-08

一、单项选择题(60分)(第小题2分，共60分)	增加本题型的题目

1.在窗体上画一个名称为Command1的命令按钮，然后编写如下事件过程：
Private Sub Command1_Click()
Move 500, 500
End Sub
程序运行后，单击命令按钮，执行的操作为
A. 命令按钮移动到距窗体左边界、上边界各500的位置
B. 窗体移动到距屏幕左边界、上边界各500的位置
C. 命令按钮向左、上方向各移动500
D. 命令按钮向左、上方向各移动500
E.
F.
答案：B
第1题的操作： 编辑 删除

二、多项选择题（30分）（每小题2共，共30分）	增加本题型的题目
三、判断题（10分）（每小题1分，共10分）	增加本题型的题目

图 16.18　显示试题的页面

```
<?
//调用格式: teach_examtest_add_singlechoice.php?examid=试卷编号
session_start();
require("../conn.php");
$examid=$_REQUEST['examid'];
$sql="select * from exam_info,course ";
$sql.="where exam_id='".$examid."' and ";
$sql.="exam_info.course_id=course.course_id";
$rs=mysql_query($sql,$conn);
$row=mysql_fetch_array($rs);
$coursename=$row['course_name'];
$exam_title=$row['exam_title'];
$exam_t_name=$row['exam_t_name'];
$sql ="select * from exam_type where exam_id=";
$sql.="'{$_REQUEST['examid']}'";
$sql.=" and exam_type_id= {$_REQUEST['exam_type_id']}";
$rs=mysql_query($sql,$conn);
$row=mysql_fetch_array($rs);
$exam_type_id=$row['exam_type_id'];
$exam_type_name=$row['exam_type_name'];
$exam_type_desc=$row['exam_type_desc'];
$auto_grade=$row['auto_grade'];
?>
<form action="teach_examtest_insert.php" method="post" enctype="multipart/
form-data" name="tk_form">
<table border="1">
```

```
    <tr> <td >课程</td><td><? echo $coursename;?></td></tr>
     <tr><td>试卷名称</td><td><? echo $exam_title;?></td></tr>
     <tr><td>题型</td>
       <td>
     <? echo $exam_type_name."(".$exam_type_desc.")";?></td>
     </tr>
     <tr><td>题目</td><td>
<table>
     <tr><td >标题</td>
     <td><textarea name="tk_content" cols="100" rows="5" ></textarea></td>
     </tr>
     <tr><td><div align="right">题目附图</div></td>
       <td><img name="pic" src="">
         <input name="tk_pic" type="file" onChange=
         "tk_form.pic.src=this.value;">
       如果该题目含有图形,则选择图片文件</td>
     </tr>
      <tr><td>选项 A.</td><td>
<textarea name="tk_option1" cols="100" ></textarea></td>
     </tr>
     <tr><td>选项 B. </td><td>
<textarea name="tk_option2" cols="100"></textarea></td>
     </tr><tr><td>选项 C. </td><td>
<textarea name="tk_option3" cols="100" ></textarea></td>
     </tr><tr><td>选项 D.</td><td>
<textarea name="tk_option4" cols="100"></textarea></td>
       </tr>
     <tr><td>选项 E. </td> <td>
<textarea name="tk_option5" cols="100"></textarea></td>
      </tr>
      <tr><td>选项 F. </td><td>
<textarea name="tk_option6" cols="100"></textarea></td>
       </tr>
       </table></td>
     </tr>
     <tr> <td>答案</td>
       <td><input type="radio" name="tk_ans" value="A"> A
         <input type="radio" name="tk_ans" value="B"> B
        <input type="radio" name="tk_ans" value="C"> C
        <input type="radio" name="tk_ans" value="D"> D
        <input type="radio" name="tk_ans" value="E"> E
        <input type="radio" name="tk_ans" value="F"> F
       </td>
     </tr>
     <tr><td>标准分值</td><td><input name="standard_score" type="text"></td>
```

```
    </tr>
    <tr><td>命题人</td> <td><? echo $exam_t_name;?></td>
    </tr>
    <tr><td colspan="2">
     <input name="examid" type="hidden" value="<? echo $examid;?>">
    <input name="tk_type_id" type="hidden" value="<? echo $exam_type_id;?>">
     <input name="auto_grade" type="hidden" value="<? echo $auto_grade;?>">
     <input name="Submit" type="submit" value="添加题目并返回">
     <input type="Submit" name="Submit" value="继续添加题目">
     <input name="Submit" type="reset" value="重新输入" >
     </td>
    </tr>
  </table>
</form>
```

运行该程序，显示的页面如图 16.19 所示，图中输入了一个单项选择题。

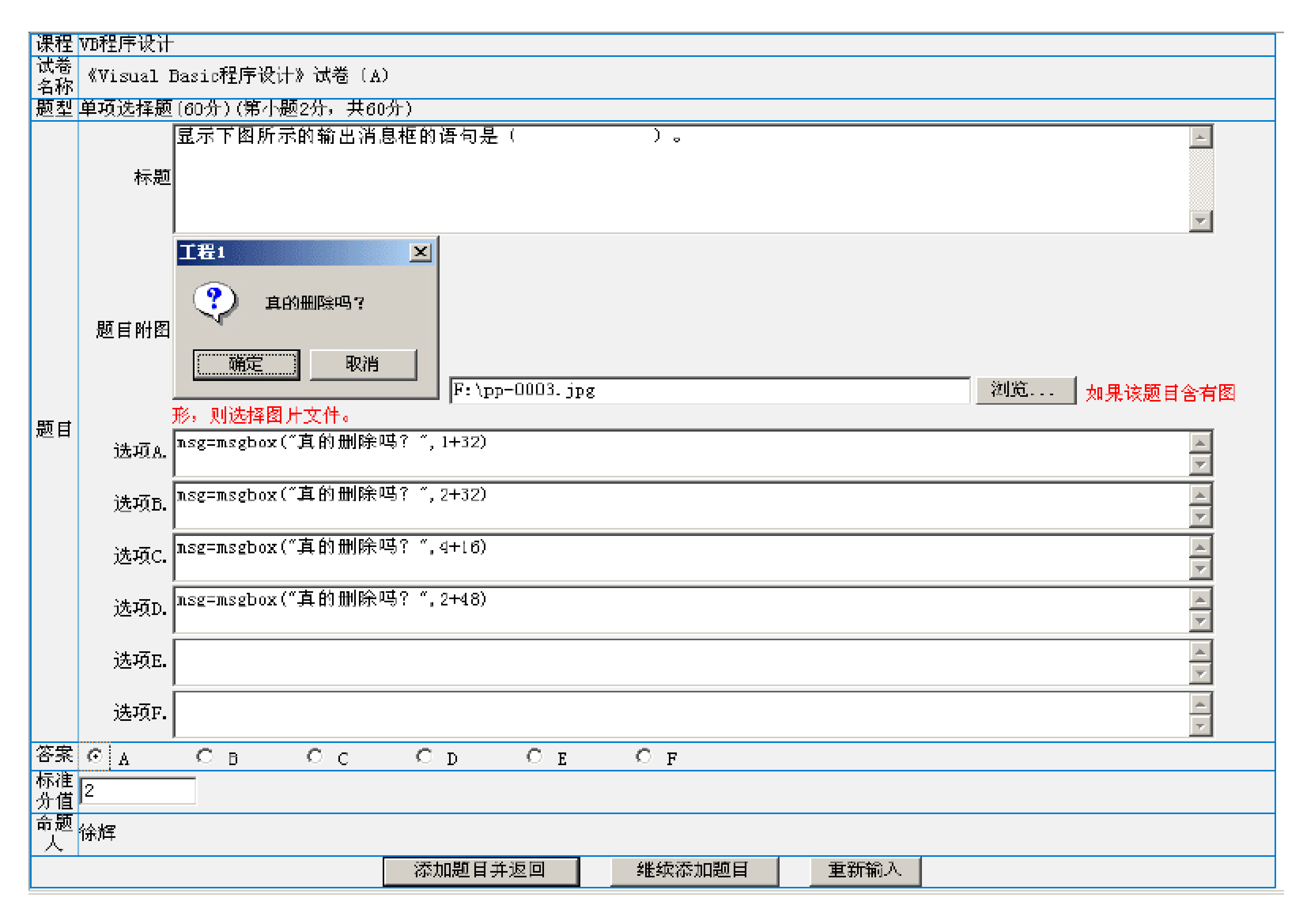


图 16.19　添加单项选择题的页面

4．增加多项选择题目的程序（teach_examtest_add_multichoice.php）

该程序根据所选试卷编号为参数，显示一个表单，输入多项选择题的题目内容、选项和答案，如果题目有插图，可以选择图形文件。该程序与增加单项选择题程序相似，不同之处是将表单的单选按钮改为复选框。

5．保存试题题目的程序（teach_examtest_insert.php）

该程序将新增的单项选择题、多项选择题和判断题型的题目内容保存到 exam_test 表。程序如下：

```
<?
//文件名: teach_examtest_insert.php
session_start();
require("../conn.php");
$examid=$_POST['examid'];
$tk_type_id=$_POST['tk_type_id'];
if ($_POST['tk_type_id']==2) {
    $tk_ans=implode(",", $_POST["tk_ans"]);
}else {
    $tk_ans=$_POST['tk_ans'];
}
//构造 INSERT 命令
if ($_POST['tk_type_id']==1 or $_POST['tk_type_id']==2) {
$sql ="INSERT INTO exam_test (exam_id,tk_type_id,";
$sql.="tk_content,tk_option1,tk_option2,tk_option3,";
$sql.="tk_option4,tk_option5,tk_option6,tk_ans,";
$sql.="standard_score,auto_grade) VALUES ";
$sql.=" ('{$_POST['examid']}',{$_POST['tk_type_id']},";
$sql.="'{$_POST['tk_content']}','{$_POST['tk_option1']}'";
$sql.=",'{$_POST['tk_option2']}','{$_POST['tk_option3']}'";
$sql.=",'{$_POST['tk_option4']}','{$_POST['tk_option5']}'";
$sql.=",'{$_POST['tk_option6']}','$tk_ans',";
$sql.="{$_POST['standard_score']},{$_POST['auto_grade']})";
}else{  //其他题型
  $sql ="INSERT INTO exam_test (exam_id,tk_type_id,";
  $sql.="tk_content,tk_ans,standard_score,auto_grade) ";
  $sql.="VALUES ('{$_POST['examid']}',";
  $sql.="{$_POST['tk_type_id']},'{$_POST['tk_content']}'";
  $sql.=",'$tk_ans',{$_POST['standard_score']},";
  $sql.="{$_POST['auto_grade']})";
}
mysql_query($sql,$conn);
$id=mysql_insert_id();
$photoname=$_FILES['tk_pic']['tmp_name'];  //上传图文件名
if (! empty($photoname)) {
  $photo=fread(fopen($photoname,"r"),filesize($photoname));
  $photo = '0x' . bin2hex($photo);
  mysql_query("update exam_test set tk_pic= $photo where st_id= $id",$conn);
}
$photoname=$_FILES['tk_ans_pic']['tmp_name'];//上传答案图文件名
if (! empty($photoname)) {
  $photo=fread(fopen($photoname,"r"),filesize($photoname));
  $photo = '0x' . bin2hex($photo);
  mysql_query("update exam_test set tk_ans_pic= $photo where st_id=
  $id",$conn);
```

```
}
if ($_POST['Submit']=="添加题目并返回") {
header("Location:teach_examtest_step2.php?examid=".$examid);
   exit;
}
if ($_POST['Submit']=="继续添加题目") {
   switch ($tk_type_id) {
      case 1:
header("Location:teach_examtest_add_singlechoice.php?examid=".$examid."
&exam_type_id=1");
           break;
       case 2:
header("Location:teach_examtest_add_multichoice.php?examid=".$examid."&
exam_type_id=2");
           break;
       case 3:
header("Location:teach_examtest_add_judge.php?examid=".$examid."&exam_t
ype_id=3");
           break;
   }
   exit;
}
?>
```

16.5.3 评阅试卷

评阅试卷的功能是根据某一试卷的考生答卷，自动地计算每个学生各题的得分，然后汇总出该考生的课程考试成绩，存储到 exam_score 表。它包括以下几个程序。

1．选择试卷程序（teach_grade_step1.php）

该程序显示要评阅的试卷名，供选择。该程序内容如下：

```
<?
session_start();
require("../conn.php");
$sql="select * from exam_time,exam_info,course ";
$sq.l="where  exam_time.exam_id=exam_info.exam_id and ";
$sq.l="exam_info.course_id=course.course_id";
$rs=mysql_query($sql,$conn) or die("查询试卷失败");
$examid=$examname=$courseid=$coursename="";
$class=$teacher=$examdate=$starttime=$endtime="";
while ($row=mysql_fetch_array($rs)) {
   $examid.='"'.$row["exam_id"].'",';
   $examtitle.='"'.$row["exam_title"].'",';
   $courseid.=$row["course_id"].',';
   $coursename.='"'.$row["course_name"].'",';
```

```
        $teacher.='"'.$row["exam_t_name"].'",';
        $examdate.='"'.$row["exam_date"].'",';
        $endtime.='"'.$row["exam_endtime"].'",';
        $starttime.='"'.$row["exam_starttime"].'",';
        $class.='"'.$row["exam_class"].'",';
    }
    if (strlen($examid)>0) {
        $examid=substr($examid,0,strlen($examid)-1);
        $examtitle=substr($examtitle,0,strlen($examtitle)-1);
        $courseid=substr($courseid,0,strlen($courseid)-1);
        $coursename=substr($coursename,0,strlen($coursename)-1);
        $teacher=substr($teacher,0,strlen($teacher)-1);
        $examdate=substr($examdate,0,strlen($examdate)-1);
        $starttime=substr($starttime,0,strlen($starttime)-1);
        $endtime=substr($endtime,0,strlen($endtime)-1);
        $class=substr($class,0,strlen($class)-1);
    }
    ?>
    <SCRIPT LANGUAGE="JavaScript">
    function select_exam(i) {
    <?
    echo "     var examid = new Array(".$examid.");\n";
    echo "     var examtitle = new Array(".$examtitle.");\n";
    echo "     var courseid = new Array(".$courseid.");\n";
    echo "     var coursename = new Array(".$coursename.");\n";
    echo "     var teacher = new Array(".$teacher.");\n";
    echo "     var examdate = new Array(".$examdate.");\n";
    echo "     var starttime = new Array(".$starttime.");\n";
    echo "     var endtime = new Array(".$endtime.");\n";
    echo "     var classids= new Array(".$class.");\n";
    ?>
        var id,i;
        timestr=examdate[i]+"  "+starttime[i]+"--"+endtime[i];
        document.form1.coursename.value=coursename[i];
        document.form1.courseid.value=courseid[i];
        document.form1.examid2.value=examid[i];
        document.form1.exam_t_name.value=teacher[i];
        document.form1.classids.value=classids[i];
        document.form1.examtitle.value=examtitle[i];
        document.all.examtime.innerHTML=timestr;
    }
    </SCRIPT>
    <body>
    <form method="POST" action="teach_grade_step2.php" name="form1">
    <table border="1">
```

```
<tr><td colspan="4" >第一步: 选择试卷</td></tr>
<tr><td>试卷名称</td> <td colspan="3">
<select name="examid" onChange="select_exam(this.selectedIndex)">
<?
$rs2=mysql_query("select * from exam_info",$conn);
while ($row2=mysql_fetch_array($rs2)) {
    echo "     <option value=\"".$row2['exam_id']."\">";
echo $row2['exam_title']."</option>\n";
}
?>
</select> </td></tr>
<tr> <td>考试课程</td>
<td colspan="3"><input name="coursename" type="text" readonly style=
"background-color:#efefef">
<input name="courseid" type="hidden">
<input name="classids" type="hidden">
<input name="examtitle" type="hidden"> </td></tr>
<tr> <td>试卷编号</td><td colspan="3">
<input name="examid2" type="text" readonly ></td> </tr>
<tr><td>命题教师</td> <td colspan="3">
<input name="exam_t_name" type="text" readonly ></td></tr>
<tr> <td >考试时间</td>
 <td colspan="3"><span id="examtime"></span> </td></tr>
</table>
<table>
<tr><td><input name="submit" type="submit" value="下一步">
</td></tr>
</table>
</form>
```

该程序的运行结果如图 16.20 所示。

2．选择考试班级程序（teach_grade_step2.php）

该程序从课程的考试班级中选择一个班级，以便下一步对该班考生进行评卷，运行的页面如图 16.21 所示（程序略）。

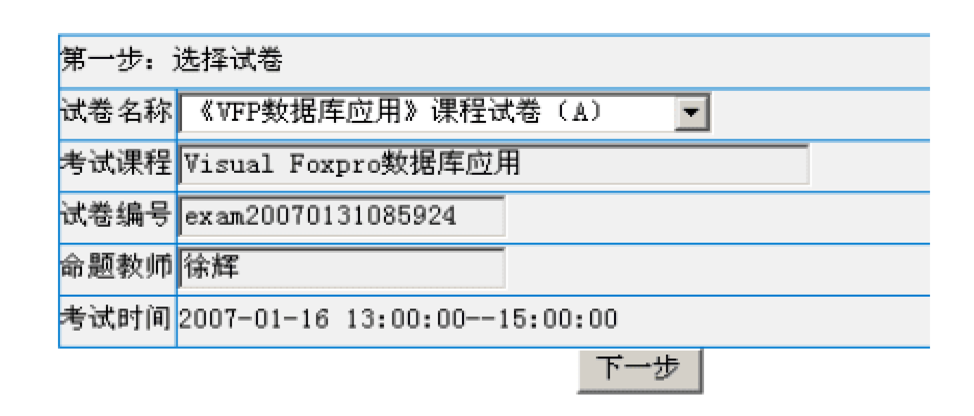


图 16.20　选择试卷页面

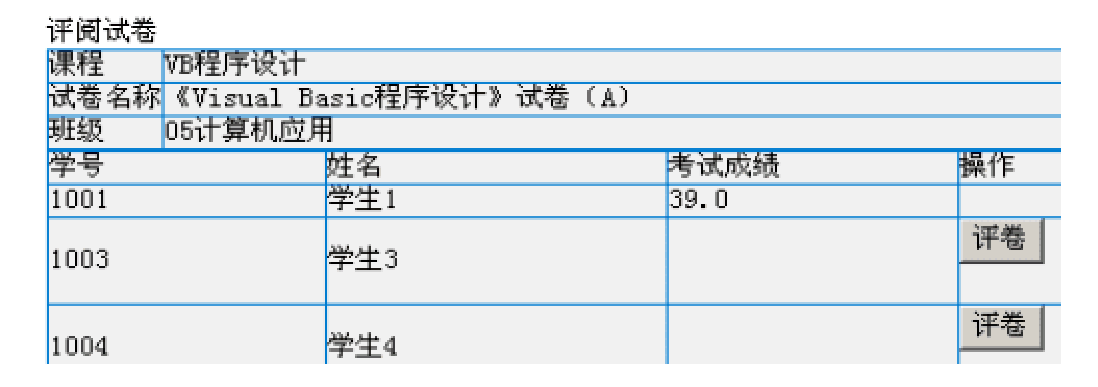
第二步：选择考试班级
考试课程　Visual Foxpro数据库应用
试卷名称　《VFP数据库应用》课程试卷（A）
考试班级　05计算机应用
下一步

图 16.21　选择考试班级页面

3．显示一个班考生成绩和操作程序（teach_grade_step3.php）

该程序根据前面所选的试卷和班级，显示一个班的课程成绩或者评卷操作。如果考生

没有成绩，则显示“评卷”按钮，如图 16.22 所示。单击“评卷”按钮，则调用评卷程序 teach_grade_step4.php，显示该考生的客观题得分情况。



评阅试卷

课程	VB程序设计
试卷名称	《Visual Basic程序设计》试卷（A）
班级	05计算机应用

学号	姓名	考试成绩	操作
1001	学生1	39.0	
1003	学生3		评卷
1004	学生4		评卷

图 16.22　成绩页面

4．评卷程序（teach_grade_step4.php）

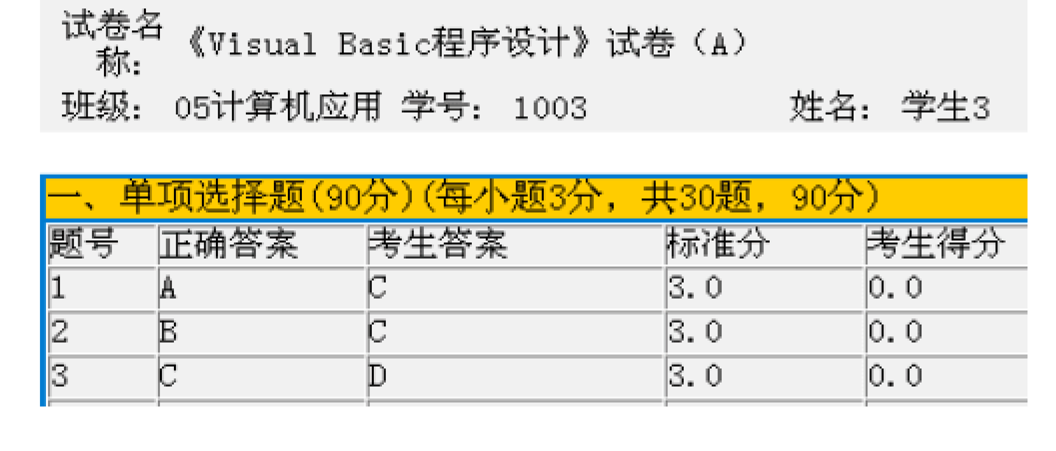
该程序根据所选择的试卷编号、班级编号和考生学号，自动地计算该考生每题的得分，并显示每题得分。程序如下：

```
<?
session_start();
require("../conn.php");
$examid=$_REQUEST['examid'];
$examtitle=$_REQUEST['examtitle'];
$xh=$_REQUEST['xh'];
$name=$_REQUEST['name'];
$coursename=$_REQUEST['coursename'];
$classname=$_REQUEST['classname'];
$sql ="update stud_exam_ans ";
$sql.="set t_userid='".$_SESSION['name']."'"
mysql_query($sql,$conn);
//给客观题自动评分
$sql ="select id,standard_score,tk_ans,stud_ans,auto_grade ";
$sql.=" from stud_exam_ans,exam_test where s_xh='$xh'";
$sql.=" and stud_exam_ans.st_id=exam_test.st_id ";
$sql.="and stud_exam_ans.exam_id='$examid' and auto_grade=1
$sql.=" order by tk_type_id";
$rs=mysql_query($sql,$conn);
while ($row=mysql_fetch_array($rs)) {
  if ($row["stud_ans"]==$row["tk_ans"]) {
      $update_sql ="update stud_exam_ans set stud_score=";
      $update_sql.=$row['standard_score']." where id=";
      $update_sql.=$row['id'];
      mysql_query($update_sql,$conn);
}else {
      $sql="update stud_exam_ans set stud_score=0 where id=";
      $sql.=$row['id'];
      mysql_query($sql,$conn);
  }
```

```
}
?>
<body >
课程名称: <? echo $coursename;?>
<br>试卷名称:  <? echo $examtitle;?>
<br>班级: <? echo $classname;?>学号:<? echo $xh;?>
姓名: <? echo $name;?><br>
<form action="teach_grade_save.php" method="post" name="form1">
<?
$sql1 ="select * from exam_type where exam_id='";
$sql1.=$examid."' order by exam_type_id";
$rs1=mysql_query($sql1,$conn) or die("查询题型错误");
$numbers="一二三";
$i=1;
while ($row1=mysql_fetch_array($rs1)) {  //外循环: 输出题型和题目
  $exam_type_id=$row1['exam_type_id'];
  $exam_type_name=$row1['exam_type_name'];
  $exam_type_desc=$row1['exam_type_desc'];
  $autograde=$row1['auto_grade'];
?>
<table border="1">
<?
//查询一个学生的某一题型的解题
  $sql2 ="select * from stud_exam_ans,exam_test where s_xh=";
  $sql2.="'$xh' and stud_exam_ans.exam_id='".$examid;
  $sql2.="' and tk_type_id=".$exam_type_id;
  $sql2.=" and stud_exam_ans.st_id=exam_test.st_id";
  $rs2 =mysql_query($sql2,$conn);
  if (mysql_num_rows($rs2)>0) {  //有该题型的题目,则显示题目
?>
  <tr ><td><font size="4">
    <? echo substr($numbers,($i-1)*2,2)."、";
      echo $exam_type_name."(".$exam_type_desc.")";?>
      </font> </td></tr>
  <tr> <td>
  <?
  if ($autograde==1) {
  ?>
  <table border="1" cellpadding="0" cellspacing="0">
    <tr> <td>题号</td><td>正确答案</td><td>考生答案</td>
      <td >标准分</td><td >考生得分</td>
    </tr>
   <?
    $sth=1;
    while ($row2=mysql_fetch_array($rs2)){ //内循环: 显示题型的题目
```

```
       ?>
      <tr> <td ><? echo $sth;?> </td>
         <td><? echo $row2['tk_ans'];?> </td>
         <td><? echo $row2['stud_ans'];?> </td>
         <td><? echo $row2['standard_score'];?> </td>
         <td><? echo $row2['stud_score'];?> </td></tr>
      <?
      $sth++;
      }  //end of while($row2...)语句
      ?>
      </table>
   <? } ?>
   </td></tr>
   <?
    $i++;
   }   // if (mysql_num_rows($rs2)>0)语句到此结束
   ?>
   </table>
   <?
   }   //外循环至此结束
   ?>
   <input name="examid" type="hidden" value="<? echo $examid;?>">
   <input name="xh" type="hidden" value="<? echo $xh;?>">
    <input type="submit" name="Submit" value="完成评卷">
   </td></tr>
   </table>
   </form>
```

访问该程序，显示的页面如图 16.23 所示。



课程名称：VB程序设计
试卷名称：《Visual Basic程序设计》试卷（A）
班级：05计算机应用 学号：1003 姓名：学生3

一、单项选择题(90分)(每小题3分，共30题，90分)

题号	正确答案	考生答案	标准分	考生得分
1	A	C	3.0	0.0
2	B	C	3.0	0.0
3	C	D	3.0	0.0

图 16.23　评卷页面

16.6　学生考试功能的程序

考生登录进入网络考试系统，参加考试、查看成绩。下面仅介绍参加考试功能的设计思想和程序实现。

进入考场功能是考生参加某一课程的考试。其设计思想是根据管理员设定的考试时间，在指定的时间内进行答题。如果考试开始时间未到，则显示剩余时间，不断测试是否到考试开始时间。当到达考试开始时间，由系统自动地将试题传输到学生端的浏览器。当考试结束时间已到，或者考生单击了“交卷”按钮，则将考生的答题内容上传服务器的数据库中存储。

进入考场采用框架结构来显示，上框架显示试题，供考生解答，框架名为 examFrame；下框架显示考试时间和剩余时间，框架名为 timeFrame。

1．进入考场框架页面主程序（stud_exam_test.php）

该程序首先检查是否已到考试开始时间。如果未到考试开始时间，则不断显示当前时间和剩余时间。如果到达考试开始时间，则进入框架页面，显示试题和考试剩余时间。程序如下：

```
<?
session_start();
require("../conn.php");
date_default_timezone_set('PRC');   //设定时区为中国时区
$cur_date=date("Y-m-d");
$cur_time=date("H:i:s");
//查询今天当前开始的时段里是否有考试课程
$sql1 ="select * from exam_time,exam_info,course ";
$sql1.="where exam_date=\"".$cur_date."\" and ";
$sql1.="(exam_starttime>=\"".$cur_time."\" or ";
$sql1.=" (exam_starttime<=\"";
$sql1.=$cur_time."\" and exam_endtime>\"".$cur_time."\")) ";
$sql1.=" and exam_time.exam_id=exam_info.exam_id and ";
  $sql1.=" exam_info.course_id=course.course_id ";
$sql1.="and exam_class like \"%".$_SESSION['class_id']."%\"";
  $rs1=mysql_query($sql1,$conn);
  if (mysql_num_rows($rs1)<=0) {
     echo "今天没有考试课程";
     exit;
  }
  //今天有考试课,测试是否到考试开始时间
  $row1=mysql_fetch_array($rs1);
  $examid=$row1['exam_id'];
  $coursename=$row1['course_name'];
  $starttime=$row1['exam_starttime'];
  $timelen=$row1['exam_timelen'];
  $endtime=$row1['exam_endtime'];
  $examdate=$row1['exam_date'];
  //考试开始时间的年、月、日、时、分、秒
  $year=substr($examdate,0,4);
  $month=substr($examdate,5,2);
  $day=substr($examdate,8,2);
```

```
$hour1=substr($starttime,0,2);
$minute1=substr($starttime,3,2);
$second1=substr($starttime,6,2);
//考试结束时间的时、分、秒
$hour2=substr($endtime,0,2);
$minute2=substr($endtime,3,2);
$second2=substr($endtime,6,2);
$startseconds=mktime($hour1,$minute1,$second1,$month,$day,$year);
$endseconds = mktime($hour2,$minute2,$second2,$month,$day,$year);
$now=time();
$now_ms=$now*1000;  //当前时间的毫秒数
$startseconds_ms=$startseconds*1000;  //考试开始时间的毫秒数
//如果未到考试开始时间,则不断测试时间,直到考试开始时间,提交表单,
//重新调用本程序 stud_exam_test.php,以便下载试题
if ($now < $startseconds) {
   $lefttime=($startseconds - $now)/60;
?>
<script language="Javascript">
hoursms=60*60*1000;
minutesms=60*1000;
secondms=1000;
//下面两行代码将服务器时间信息传递给客户机脚本程序
nowTime=new Date(<? echo $now_ms;?>);
startTime=new Date(<? echo $startseconds_ms;?>);
diffms=startTime.getTime()-nowTime.getTime();
function timeCount()
{
   totalms=diffms;
   leftHours=Math.floor(totalms /hoursms);
   totalms -= leftHours *hoursms;
   leftMinutes=Math.floor(totalms/minutesms);
   totalms -= leftMinutes * minutesms;
   leftSeconds=Math.floor(totalms /secondms);
   diffms-=1000;
  timestr=leftHours+"小时"+leftMinutes+"分"+leftSeconds+"秒";
   if (leftHours==0 && leftMinutes==0 && leftSeconds==0) {
      document.timeform.submit();
   } else {
      document.all.time1.innerHTML=timestr;
   }
   setTimeout("timeCount()",1000);
}
window.onload=timeCount;
</script>
<  title>服务器倒计时</title></head><body>
```

```
<?
echo "考试课程: ".$coursename;
echo "<br>试卷编号: ".$row1['exam_id'];
echo "<br>试卷名称: ".$row1['exam_title'];
echo "<br>考试日期: ".$examdate;
echo "<br>考试时间: ".$starttime."--".$endtime." 共(".$timelen.")分钟";
?>
<form action="stud_exam_test.php" method="post" name="timeform" >
<table border="0" cellpadding="2" cellspacing="2">
  <tr><td>未到考试开始时间,距离考试开始时间,还剩下</td>
    <td><span id="time1"></span></td><td>请稍候...</td></tr>
</table>
</form></body>
<?
   exit;
} else if ($now < $endseconds) {
?>
<html><head><title>网上考试</title></head>
<frameset rows="95%,*" cols="*" framespacing="0" border="0">
  <frame src="stud_test_display.php?call=stud_test_display.php&examid=
  <? echo $examid;?>" name="examFrame" scrolling="yes">
  <frame src="time_count_to_zero.php" name="timeFrame" scrolling="no">
</frameset><noframes><body></body></noframes></html>
<?
  exit;
}
?>
```

访问该程序，显示的页面有图 16.24 和图 16.25 两种可能的页面。

考试课程：VB程序设计
试卷编号：exam20070109100649
试卷名称：《Visual Basic程序设计》试卷（A）
考试日期：2007-01-11
考试时间：09:00:00--11:00:00 共(120)分钟

未到考试开始时间，距离考试开始时间,还剩下 0小时39分15秒 请稍候...

图 16.24　考试时间未到的页面

2．考试时间倒计时程序（time_to_0.php）

该程序根据 exam_time 表中当天考试时间的安排，显示当前考试时间和考试剩余时间，参见图 16.25。一旦考试时间结束，通过调用 JavaScript 脚本程序，自动地提交上框架的考生答卷内容，传送到 Web 服务器。程序如下：

```
<?
session_start();
require("../conn.php");
```

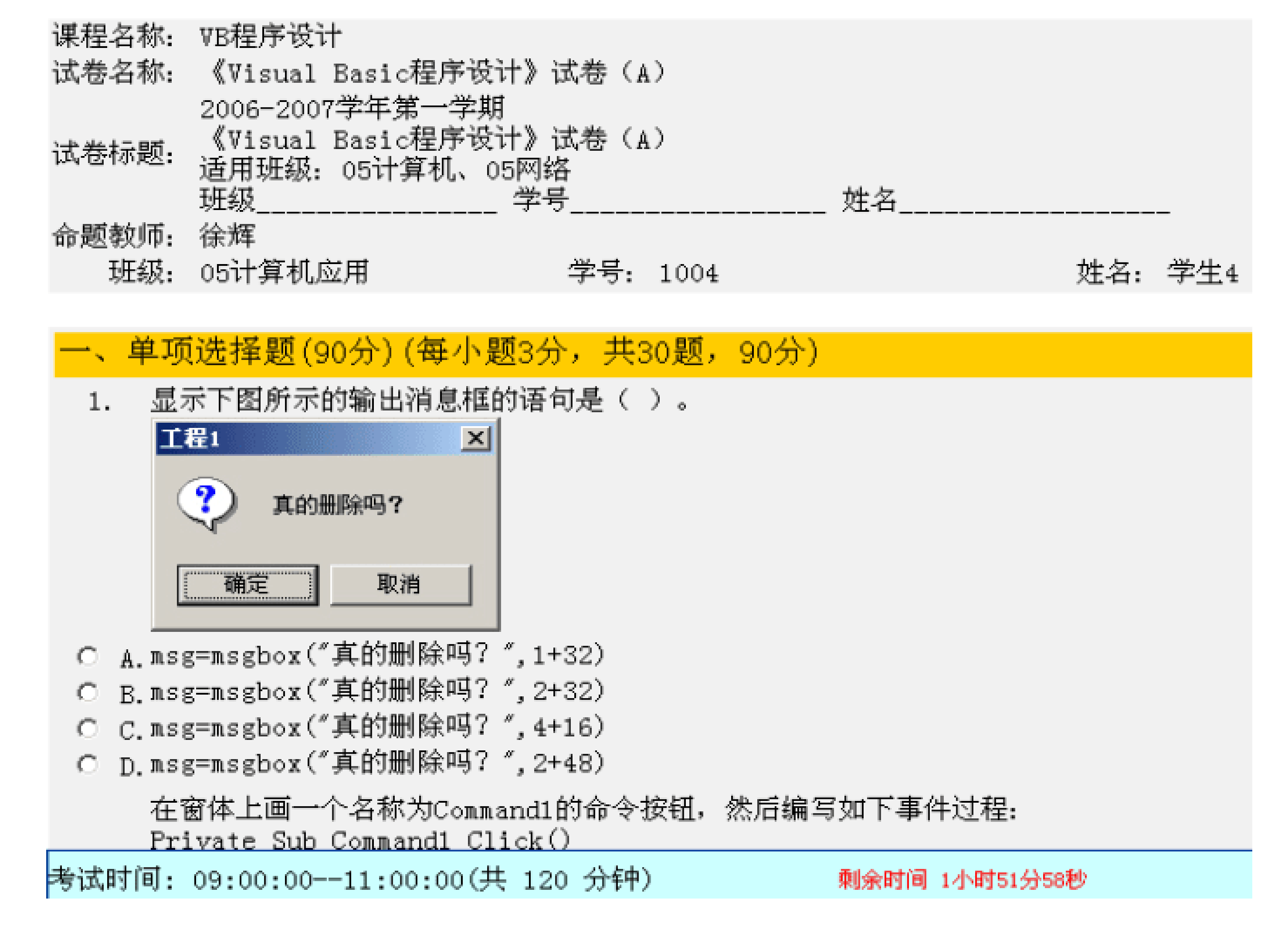


图 16.25　开始考试的页面

```
$cur_date=date("Y-m-d");
$cur_time=date("H:i:s");
$sql1 ="select * from exam_time,exam_info,course where ";
$sql1.=" exam_date=\"".$cur_date."\" and ";
$sql1.="(exam_starttime>=\"".$cur_time."\" or ";
$sql1.="(exam_starttime<=\"".$cur_time."\" and "
$sql1.="exam_endtime>\"".$cur_time."\")) ";
$sql1.=" and exam_time.exam_id=exam_info.exam_id and
$sql1.=" exam_info.course_id=course.course_id ";
$sql1.=" and exam_class like \"%".$_SESSION['class_id']."%\"";
$rs1=mysql_query($sql1,$conn);
$row1=mysql_fetch_array($rs1);
$examid=$row1['exam_id'];
$coursename=$row1['course_name'];
$starttime=$row1['exam_starttime'];
$timelen=$row1['exam_timelen'];
$endtime=$row1['exam_endtime'];
$examdate=$row1['exam_date'];
$year=substr($examdate,0,4);
$month=substr($examdate,5,2);
$day=substr($examdate,8,2);
$h1=substr($starttime,0,2);
$m1=substr($starttime,3,2);
$s1=substr($starttime,6,2);
$h2=substr($endtime,0,2);
```

```
  $m2=substr($endtime,3,2);
$s2=substr($endtime,6,2);
$startseconds=mktime($h,$m1,$s1,$month,$day,$year);
$endseconds=mktime($h2,$m2,$s2,$month,$day,$year);
$exam_minutes=($endseconds - $startseconds)/60;
$now=time();                          //当前时间的秒数
$now_ms=$now*1000;                    //当前时间的毫秒数
$endseconds_ms=$endseconds *1000;     //考试结束时间的毫秒数
?>
<script language="Javascript">
hoursms=60*60*1000;
minutesms=60*1000;
secondms=1000;
nowTime=new Date(<? echo $now_ms;?>);
startTime=new Date(<? echo $endseconds_ms;?>);
diffms=startTime.getTime()-nowTime.getTime();
function timeCount()
{
   totalms=diffms;
   leftHours=Math.floor(totalms /hoursms);
   totalms -= leftHours *hoursms;
   leftMinutes=Math.floor(totalms/minutesms);
   totalms -= leftMinutes * minutesms;
   leftSeconds=Math.floor(totalms /secondms);
   diffms-=1000;
   tstr=leftHours+"小时"+leftMinutes+"分"+leftSeconds+"秒";
   if (leftHours==0 && leftMinutes==0 && leftSeconds==0) {
        parent.examFrame.document.form1.submit();
        alter("考试时间已到,系统自动交卷!");
   } else {
     document.all.time2.innerHTML=tstr;
   }
   setTimeout("timeCount()",1000);
}
window.onload=timeCount;
</script>
<table border="0" ><tr><td>考试时间:
<? echo $starttime."--".$endtime."(共 $exam_minutes 分钟)";?>
 </td><td><div style="font-size:9pt;color:red">剩余时间
<span id="time2"></span></div></td>
 <td width="15%"> </td>
 </tr>
</table>
```

实　验　16

1．上机验证网络考试系统的各个程序。

2．扩展网络考试系统的功能，要求每个学生的账户只能在一台计算机上登录，不允许同时在多台计算机上登录系统。

参 考 文 献

1　王军等译. MySQL4 从入门到精通. 北京：电子工业出版社，2003
2　梁志敏等译. 数据库驱动的 Web 站点开发（第 2 版）. 北京：清华大学出版社，2003
3　袁勤勇等译. Apache Server 2.0 技术参考大全. 北京：清华大学出版社，2003
4　陈德华等译. Apache 管理员手册. 北京：机械工业出版社，2003
5　郭金锋，林宇等. PHP & MySQL Web 网络编程. 北京：人民邮电出版社，2001
6　W Jason Gilmore. Beginning PHP and MySQL 5：From Novice to Professional. Apress，2006
7　邓云佳等译. PHP 程序设计. 北京：中国电力出版社，2003
8　Robert Richards. Pro PHP XML and Web Services.Apress，2006
9　Paul Reinheimer. Professional Web APIs with PHP.Wrox Press，2006
10　贺民等译.PHP 技术内幕. 北京：中国水利水电出版社，2003
11　Julie Meloni. Sams Teach Yourself PHP, MySQL and Apache All in One. Sams Publishing，2006
12　Lee Babin.PHP 5 Recipes: A Problem-Solution Approach. Apress，2005
13　四维科技. PHP 网络编程技术与实例. 北京：人民邮电出版社，2006
14　王石,杨英娜. 精通 PHP+MySQL 应用开发. 北京：人民邮电出版社，2006
15　刘猛. Web 应用程序开发. 北京：高等教育出版社，2004
16　[美] Aaron Skonnard, Martin Gudgin 著. 牛韬，英宇译. XML 精要参考手册. 北京：人民邮电出版社，2002
17　武晓军，陈海滨.JavaScript/VBScript 网页编程实例解析. 北京：清华大学出版社，2001
18　[美]Danny Goodman 著. 汪厚祥等译. JavaScript 宝典. 北京：电子工业出版社，1999
19　姜晓铭等. PHP 程序设计与实例分析教程. 北京：清华大学出版社，2001
20　李香敏. PHP MySQL Apache 超强组合. 西安:西安电子科技大学出版社，2001
21　程向前.基于开放平台的网页设计与编程. 北京：清华大学出版社，2002
22　郝兴伟. Web 技术导论. 北京：清华大学出版社，2005
23　[美]John Pollock 著. 云巅工作室译. JavaScript 编程起步. 北京：人民邮电出版社，2001
24　徐辉. 基于 XML-RPC 和 PHP5 的 Web 服务的研究和实现. 福建电脑，2005（10）
25　徐辉. 基于 PHP 的 Web 报表输出的设计与实现. 现代图书情报技术，2006（年刊）
26　徐辉. 基于 Google Web 服务和 PHP 的搜索引擎的设计与实现. 福建电脑，2007（3）

“21 世纪高等学校计算机教育实用规划教材”系列书目

书　　名	作　　者	ISBN 号
32 位微型计算机原理・接口技术及其应用（第 2 版）	史新福等	9787302134039
AutoCAD 实用教程（配光盘）	张强华等	9787302127260
Internet 实用教程——技术基础及实践	田力	9787302110668
Java 程序设计实践教程	张思民	9787302132585
Java 程序设计实用教程	胡伏湘等	9787302109600
Java 语言程序设计	张思民	9787302144113
Visual Basic 程序设计基础	李书琴等	9787302132684
XML 实用技术教程	顾兵	9787302142867
大学计算机公共基础	阮文江	9787302143307
大学计算机网络公共基础教程	徐祥征等	9787302130161
多媒体技术教程——案例、训练与课程设计	胡伏湘等	9787302126201
多媒体课件制作——Authorware 实例教程	唐前军等	9787302156000
汇编语言程序设计教程与实验	徐爱芸	9787302143413
计算机操作系统	颜彬等	9787302141471
计算机网络实用教程——技术基础与实践	刘四清等	9787302104513
计算机网络应用技术教程	孙践知	9787302118893
计算机网络与 Internet 实用教程——技术基础与实践	徐祥征等	9787302106593
实用软件工程	陆惠恩	9787302125594
数据库及其应用系统开发（Access 2003）	张迎新	9787302128281
数据库技术与应用——SQL Server	刘卫国等	9787302143673
数据库技术与应用实践教程——SQL Server	严晖、刘卫国	9787302142317
数据库应用案例教程（Access）	周安宁 张新猛等	9787302146056
网络技术应用教程	梁维娜等	9787302134848
网页制作教程	夏宏等	9787302105916
微型计算机原理及应用导教・导学・导考（第 2 版）	史新福等	9787302133995
程序设计语言——C	王珊珊等	9787302158035

读者意见反馈

亲爱的读者：

感谢您一直以来对清华版计算机教材的支持和爱护。为了今后为您提供更优秀的教材，请您抽出宝贵的时间来填写下面的意见反馈表，以便我们更好地对本教材做进一步改进。同时如果您在使用本教材的过程中遇到了什么问题，或者有什么好的建议，也请您来信告诉我们。

地址：北京市海淀区双清路学研大厦 A 座 602 室　　计算机与信息分社营销室 收
邮编：100084　　　　电子邮箱：jsjjc@tup.tsinghua.edu.cn
电话：010-62770175-4608/4409　　邮购电话：010-62786544

教材名称：PHP Web 程序设计教程与实验
ISBN：978-7-302-15550-8

个人资料

姓名：＿＿＿＿＿＿年龄：＿＿＿所在院校/专业：＿＿＿＿＿＿＿＿

文化程度：＿＿＿＿通信地址：＿＿＿＿＿＿＿＿＿＿＿＿＿＿

联系电话：＿＿＿＿电子信箱：＿＿＿＿＿＿＿＿＿＿＿＿＿＿

您使用本书是作为： □指定教材　□选用教材　□辅导教材　□自学教材

您对本书封面设计的满意度：

□很满意　□满意　□一般　□不满意　改进建议＿＿＿＿＿＿＿＿

您对本书印刷质量的满意度：

□很满意　□满意　□一般　□不满意　改进建议＿＿＿＿＿＿＿＿

您对本书的总体满意度：

从语言质量角度看　□很满意　□满意　□一般　□不满意

从科技含量角度看　□很满意　□满意　□一般　□不满意

本书最令您满意的是：

□指导明确　□内容充实　□讲解详尽　□实例丰富

您认为本书在哪些地方应进行修改？（可附页）

＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

您希望本书在哪些方面进行改进？（可附页）

＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

电子教案支持

敬爱的教师：

为了配合本课程的教学需要，本教材配有配套的电子教案（素材），有需求的教师可以与我们联系，我们将向使用本教材进行教学的教师免费赠送电子教案（素材），希望有助于教学活动的开展。相关信息请拨打电话 010-62776969 或发送电子邮件至 jsjjc@tup.tsinghua.edu.cn 咨询，也可以到清华大学出版社主页（http://www.tup.com.cn 或 http://www.tup.tsinghua.edu.cn）上查询。